A data driven black box approach for the inverse quantification of set-theoretical uncertainty

Lars Bogaerts
Department of Mechanical Engineering
LMSD division
KU Leuven
Jan De Nayerlaan 5
St.-Katelijne-Waver, 2860, Belgium
Flanders Make@Kuleuven
Email: lars.bogaerts[at]kuleuven.be

Matthias Faes
Chair for Reliability Engineering
TU Dortmund University
Leonhard-Euler-Strasse 5
Dortmund, 44227, Germany
Email: matthias.faes[at]tu-dortmund.de

David Moens
Department of Mechanical Engineering
LMSD division
KU Leuven
Jan De Nayerlaan 5
St.-Katelijne-Waver, 2860, Belgium
Flanders Make@Kuleuven
Email: david.moens[at]kuleuven.be

ABSTRACT

Inverse uncertainty quantification commonly uses the well established Bayesian framework. Recently, alternative interval methodologies have been introduced. However, in their current state of the art implementation, both techniques suffer from a large and usually unpredictable computational effort. Thus, both techniques are not applicable in a real-time context. To achieve a low-cost, real-time solution to this inverse problem, we introduce a deep-learning framework consisting of unsupervised autoencoders and a shallow neural network. This framework is trained by means of a numerically generated dataset that captures typical relations between the model parameters and selected measured system responses. The performance and efficacy of the technique is illustrated using two distinct case studies. The first case involves the DLR AIRMOD, a benchmark case that has served as reference case for the inverse uncertainty quantification problem. The results demonstrate that the achieved accuracy is on par with the existing interval method found

in literature, while requiring only a fraction of its computational resources. The second case study examines a resistance pressure welding process, which is known to require extremely fast monitoring and control due to the high process throughput. Based on the proposed method, and with only a limited selection of simulated responses of the process, it is possible to identify the interval uncertainty of the crucial parameters of the process. The computational cost in this case makes it possible for an inverse uncertainty quantification in a real-time setting.

1 INTRODUCTION

Over the past decades, there has been growing interest in the development and utilisation of numerical modelling techniques. There is good reason for this, as the ever rising computational power is capable of supporting state-of-the-art techniques, often involving many degrees of freedoms, highly non-linear behaviour and complicated sets of governing equations. However, the information of only one deterministic numerical model is rather limited. After all, the knowledge on a structure or process is usually incomplete, be it due to inherent variable parameters or a lack of knowledge. Therefore, approaches for uncertainty quantification are required to cope with the considerable uncertainty stemming from diverse factors such as variable model parameters, boundary conditions, geometric variables or a lack of knowledge on specific parameter values. To accommodate these variabilities and uncertainties, several philosophies are widely adopted, including probabilistic techniques, [1], interval methods[2] or imprecise probabilistic frameworks[3]. For many applications, e.g., process control or reliability estimation of a system, capturing the non-deterministic parameters of a model is key for a realistic quantification of the non-determinism in the responses of the model. However, not all parameters are trivial to measure (e.g., heterogeneous material properties) or are unobservable, e.g., due to temporal effects (e.g. temporal (in)stability). To identify and quantify those non-deterministic parameters of a model that cannot be measured directly, inverse uncertainty quantification (UQ) techniques have been introduced. Following inverse UQ, the responses stemming from the structure or process are measured and used to infer knowledge on the non-determinism in the model parameters. In the context of inverse UQ, both probabilistic and non-probabilistic approaches have been developed, and practically compared in terms of computational efficiency, conservatism in encompassing the dataset and required assumptions on priors information [4, 5]. The probabilistic method, where the class of Bayesian methods is considered the standard approach, starts from assigning a likelihood to the uncertain parameters of the numerical model, and are aimed at inferring the relative likelihood of possible realisations of the model responses [6]. On the other hand, set-based interval methods define boundaries that encompass

non-deterministic parameters, but they lack the capability to assign a likelihood to each value within this range. Elaborate literature exists on both techniques, comparing these concepts from a theoretical and implementation point of view (see, e.g., [7, 8]).

It can be concluded that both methods are very valuable to provide either a tight estimation of the distribution of, or at least bounds on, the uncertainty quantities under consideration. The underlying philosophy differs greatly and the most appropriate method given an inverse UQ problem is highly case dependent. Both approaches posses strengths and weaknesses which have to be considered, e.g., availability of the data, desired information on the uncertainty, available computational power vs. required computational effort for the specific problem.

This paper specifically focuses on interval approaches, as we want to focus on the specific case where only very few data replica are available. Due to the excessive computational cost of inverse approaches, they are typically not suitable for real-time applications. A possible solution to this problem is to replace the expensive numerical model with an accurate surrogate model that is faster to evaluate [9]. A second main problem lies in the high dimensionality of datasets commonly found in industrial applicable models [10]. This high dimensionality induces challenges, particularly for an inverse interval quantification, where the calculation of convex hulls is necessary. Due to the exponential scaling of computational cost with dimensionality, reducing dimensionality is crucial to maintain feasible computation times. This requires tools from the domains of big data & machine learning [11, 12]. Various methods can be employed, e.g., based on covariance matrix decompositions, active subspace methods or manifold learning. Also autoencoders, as introduced by Olshausen and Field [13], are increasingly being exploited in modern machine- and deep-learning techniques [14]. However, even when these approaches are being considered, the computational cost that is associated with solving the inverse interval problem using the methods discussed in [15] might be excessive. This paper presents a combined unsupervised and supervised deep learning approach for the inverse quantification of set-theoretical uncertainty. The core idea is to construct a neural network as an inverse surrogate model. This is achieved by training the model using data generated by a numerical code representing the forward analysis. This paper is an extension on the work by the authors in [16]. The method is applied on two independent cases.

The first one is a benchmark case on this topic, namely the DLR AIRMOD test setup, and its corresponding dataset, which are described in detail in [17]. The second case is based on a dataset stemming from a self developed validated Finite Element model of a resistance pressure welding (RPW) process.

These cases are selected due their challenging nature and relevance on the topic. The selection of

the first case is based on its extensive coverage in the literature, featuring discussions on the applicability of Bayesian and interval methods in scenarios with both abundant and scarce data [4]. The second case is a highly relevant topic for industry, since real-time process control on the RPW process is challenging to achieve, but is crucial to guard the process quality. The process is based on a set of nominal process parameters, generally accompanied by very restrained controlling schemes, and is often hampered by minor perturbations, providing significant varying results. Therefore, identifying the uncertainty in an inverse setting provides insight for appropriate control actions.

This paper is structured as follows: section 2 elaborates the aforementioned current state-of-the-art on the inverse interval uncertainty quantification (iUQ) approach. Section 3 discusses the general methodology of applying a black box model in an inverse uncertainty quantification method. Section 4 and section 5 illustrate the performance of the this methodology on two distinctive cases. Finally, section 6 lists the conclusions of this work.

2 INVERSE QUANTIFICATION OF MULTIVARIATE INTERVAL UNCERTAINTY

This section contains an introduction to the interval approach for the inverse identification of uncertainty. The method is briefly discussed in this work to provide the reader an insight into the state-of-the-art in inverse interval methods. Furthermore, a practical comparison of the method with the multivariate interval approach is presented in the context of a case study. For a more in depth study on the latter, the most relevant literature according to the authors is given at relevant points in the summary.

An inverse method arises when parameters are unobservable or cannot be measured, and have to be estimated based on measured responses of the system, which are only indirectly related to the parameters though a computational model $\mathcal{M}$. For simplicity in this work, vectors are expressed as lowercase boldface characters $\boldsymbol{\theta}$. The numerical model is parameterised by a parameter vector $\boldsymbol{\theta} \in \mathcal{F} \subset \mathbb{R}^k$, with $\mathcal{F}$ the set of physically admissible parameters and $k \in \mathbb{N}^+$. Evaluating the model $\mathcal{M}(\boldsymbol{\theta})$ for a certain realisation of $\boldsymbol{\theta}$ yields a vector of model responses $\boldsymbol{z}^s \in \mathcal{H} \subset \mathbb{R}^d$, with $d \in \mathbb{N}^+$ and $\mathcal{H}$ the set of physically admissible model responses, according to:

$$\boldsymbol{z}^s = \mathcal{M}(\boldsymbol{\theta}). \tag{1}$$

The problem statement in this paper is called inverse, because instead of propagating parameters through a model, the goal is to use measurement data to obtain insight on the model parameters, which constitutes an inverse problem.

In an interval context, a model parameter θ subject to uncertainty is denoted as $\boldsymbol{\theta}^I$. By definition, the index I is used to indicate an interval valued parameter. The possible values in the set $\boldsymbol{\theta}^I$ are constrained by both an upper bound $\overline{\theta}$ and a lower bound $\underline{\theta}$, where $\underline{\theta} \leq \overline{\theta}$. An interval is closed when both bounds are a member of the interval. The domain of closed real-valued intervals is denoted as $\mathbb{R}$. Within $\mathbb{R}$, an interval $\boldsymbol{\theta}^I$ is explicitly defined as:

$$\boldsymbol{\theta}^I = \left[\underline{\theta}, \overline{\theta}\right] = \left\{\theta \in \mathbb{R} \,|\, \underline{\theta} \leq \theta \leq \overline{\theta}\right\}. \tag{2}$$

Note that an interval can equivalently be represented as a closed section of the real line. An interval

vector is a vector with each element a real-valued interval:

$$\boldsymbol{\theta}^{\mathbf{I}} = \begin{Bmatrix} \boldsymbol{\theta}_1^I \\ \boldsymbol{\theta}_2^I \\ \vdots \\ \boldsymbol{\theta}_k^I \end{Bmatrix} = \left\{ \boldsymbol{\theta} \in \mathcal{F} \subset \mathbb{R}^k \,|\, \theta_i \in \theta_i^I \right\}, \tag{3}$$

with $\boldsymbol{\theta}^{\mathbf{I}} \subseteq \mathbb{R}^k$, the domain of closed real-valued interval vectors of dimension k and $\mathcal{F}$ the sub-domain of feasible parameters (e.g., non-negative material properties). Analogue to interval vectors, interval matrices are defined on $\mathbb{R}^{k \times m}$.

2.1 Optimisation based approach

Consider $\boldsymbol{\theta}^I$ as the interval vector containing the k uncertain input parameters, while $\mathcal{M}(\boldsymbol{\theta})$ is a model as described as in Eq. (1). The interval FE method can then be expressed as finding the solution set $\tilde{\boldsymbol{z}}$:

$$\tilde{\boldsymbol{z}} = \left\{ \boldsymbol{z} \mid \boldsymbol{z} = \mathcal{M}(\boldsymbol{\theta}), \forall \boldsymbol{\theta} \in \boldsymbol{\theta}^I \right\}, \tag{4}$$

which reads as "$\tilde{\boldsymbol{z}}$ is the realisation set comprising all output vectors $\boldsymbol{z}$, obtained through the deterministic application of a numerical procedure to all vectors $\boldsymbol{\theta}$, contained in $\boldsymbol{\theta}^I$". It is important to note that the result $\tilde{\boldsymbol{z}}$ of an interval FE computation is typically not represented as an interval vector. As per definition, all entries in an interval vector are decoupled and are therefore not capable of describing dependency between different variables. However, the model structure of $\mathcal{M}$ (e.g., in the form of sets of differential equations) provides a coupling among all model responses $\boldsymbol{z}$. Consequently, representing the result with an interval vector may include non-physical model response vectors $\boldsymbol{z}$ explicitly, potentially causing a significant degree of conservatism. Moreover, in general, $\tilde{\boldsymbol{z}}$ spans a non-convex manifold in $\mathbb{R}^d$. As such, a closed form solution of $\tilde{\boldsymbol{z}}$ can only be obtained when an explicit analytical solution for $\boldsymbol{z} = f(\boldsymbol{\theta})$ exists.

As an alternative approach, the exact solution set $\tilde{\boldsymbol{z}}$ can also be approximated by the construction of an uncertain realization set $\tilde{\boldsymbol{z}}_s$ under e.g., convexity assumptions. This set is obtained by calculating q

deterministic responses $\boldsymbol{z}_j^s$ of by propagating well-chosen $\boldsymbol{\theta}_f \in \boldsymbol{\theta}^I, f = 1, \ldots, q$:

$$\tilde{\boldsymbol{z}}^s = \left\{ \boldsymbol{z}_j^s \mid \boldsymbol{z}_j^s = \mathcal{M}(\boldsymbol{\theta}_f), \boldsymbol{\theta}_f \in \boldsymbol{\theta}^I; f = 1, \cdots, q \right\}, \tag{5}$$

with $\boldsymbol{z}_j^s \in \mathbb{R}^d, j = 1, ..., q$ a vector containing the d-dimensional output response of the j^{th} deterministic model solution:

$$\boldsymbol{z}_j^s = [z_1^s, z_2^s, ..., z_d^s]^T, \tag{6}$$

In practice, the response quantities that constitute $\boldsymbol{z}_j^s$ depend on the considered model $\mathcal{M}$. It is imperative that the realisations represent the solution set $\tilde{z}$ as closely as possible, under condition that the realisations remain within the extreme model responses defined by $\mathcal{M}(\boldsymbol{\theta}^{\mathbf{I}})$. Therefore, recent work has focused on finding the smallest conservative convex approximation of $\tilde{z}$ [15]. Other approaches involve sampling from a Cauchy distribution [18], or Bayesian optimisation schemes [19]

When $\mathcal{M}$ is a strictly monotonic FE model, the Transformation Method [20] provides an exact mapping from $\boldsymbol{\theta}^{\mathbf{I}}$ to $\tilde{z}$, albeit needing 2^k deterministic model evaluations for the solution of a single interval problem. A mathematical handle to the boundaries of $\tilde{\boldsymbol{z}}^s$ is provided by the convex hull $\mathcal{C}^s$ of the realisation set $\tilde{\boldsymbol{z}}^s$. $\mathcal{C}^s$ can be considered as a set of vertices bounding the uncertain realisation set. This set of vertices in essence equivalently represent a convex polytope in $\mathbb{R}^d$ according to the Minkowski-Weyl theorem. As such, a convex hull can be endowed with a Lebesgue measure $\mathcal{V}_s$ on $\mathbb{R}^d$. For inverse quantification of multivariate interval uncertainty, experimental data $\mathcal{D}^e$ of the model responses $\boldsymbol{z}$ are used to construct a measurement set $\tilde{\boldsymbol{z}}^e$:

$$\tilde{\boldsymbol{z}}^e = \left\{ \boldsymbol{z}_j^e \in \mathbb{R}^d | j = 1, \cdots, q \right\}, \tag{7}$$

with $\boldsymbol{z}_j^e \in \mathbb{R}^d, j = 1, ..., q$ a vector containing the d-dimensional parameters of the j^{th} deterministic experiment. The non-determinism that is present in this set is equivalently bounded by a convex hull $\mathcal{C}^e$, with the corresponding measure $\mathcal{V}_e$.

Finally, the interval uncertainty in $\tilde{\boldsymbol{\theta}}^e_{MViUQ}$is identified by minimising the following objective function:

$$\tilde{\boldsymbol{\theta}}^e_{MViUQ} = \operatorname{argmin} \delta(\boldsymbol{\theta}^I) = (\Delta V_s^2 + w_0 \Delta V_0^2 + \Delta c^2)^{1/2}, \tag{8}$$

which includes the term ΔV_s as a measure for the difference in the measures of the experimental $\mathcal{V}_e$ and simulated $\mathcal{V}_s$ convex hulls $\mathcal{C}^e$ and $\mathcal{C}^s$, ΔV_0 a measure for the overlap in both convex hulls and Δc a measure for the difference in location of the centre of gravity of both convex hulls. The weighting factor w_0 is initially set to unity, which balances the priority between maximising the overlap and matching the multidimensional volumes. This also serves to mitigate instabilities. These would inevitably manifest in the case a large w_0 is selected due to the discontinuous behaviour of Eq. (8) as a result of the barrier-function-like term $w_0 \Delta V_0^2$. The index $_{MViUQ}$ refers to the optimisation based approach in this subsection, and serves as a notation during the comparison in the following section 4. For more details on this approach, the reader is referred to either [15] or [21]. This method serves as the reference for our approach introduced in this work.

2.2 Point-wise inverse approach

The identification procedure in the proposed point-wise inversion process starts from the training of a learning model $\mathcal{G}$ that represents the inverse of the forward model $\mathcal{M}$. The objective of this surrogate is to enable the direct calculation of the approximated interval input parameter space $\tilde{\boldsymbol{\theta}}^e_{BB}$, with the index $_{BB}$ referring to the point-wise black box learning approach in this subsection, and serves as a notation during the comparison in the following section 4. The direct calculation is based on the set of measured responses $\tilde{\boldsymbol{z}}^e$ coming from an experimental data set $\mathcal{D}^e$. The set $\tilde{\boldsymbol{\theta}}^e_{BB}$ is obtained by inversely propagating q deterministic measurements $\boldsymbol{z}^e_j$, with $j = 1, \cdots, q$ to the corresponding input trough the surrogate. Specifically, we aim to represent the relation between $\boldsymbol{\theta}^e$ and $\boldsymbol{z}^e$ as:

$$\boldsymbol{\theta}^e_j = \mathcal{G}(\boldsymbol{z}^e_j), \tag{9}$$

with $\mathcal{G}$ referring to the inverse model of a classic forward computational model $\mathcal{M}$, that is calculated q times with $j = 1, \cdots, q$. The set of approximated interval input parameters $\tilde{\boldsymbol{\theta}}^e_{BB}$ is obtained by defining the bounds stemming from the point-wise approach for each model parameter $\boldsymbol{\theta}^e_{BB}(i)$:

$$\tilde{\boldsymbol{\theta}}^e(i) = \left[\underline{\boldsymbol{\theta}}^e(i), \overline{\boldsymbol{\theta}}^e(i)\right] \tag{10}$$

where:

$$\underline{\boldsymbol{\theta}}^e(i) = \min_{j=1,\dots q} \boldsymbol{\theta}_j^e(i) \tag{11}$$

$$\overline{\boldsymbol{\theta}}^e(i) = \max_{j=1,\dots q} \boldsymbol{\theta}_j^e(i) \tag{12}$$

which constructs the set $\tilde{\boldsymbol{\theta}}^e$ for d parameters, where $i = 1, \cdots, d$:

$$\tilde{\boldsymbol{\theta}}^e = \left\{ \begin{array}{c} \left[\underline{\boldsymbol{\theta}}^e(1), \overline{\boldsymbol{\theta}}^e(1)\right] \\ \vdots \\ \left[\underline{\boldsymbol{\theta}}^e(i), \overline{\boldsymbol{\theta}}^e(i)\right] \\ \vdots \\ \left[\underline{\boldsymbol{\theta}}^e(d), \overline{\boldsymbol{\theta}}^e(d)\right] \end{array} \right\} \tag{13}$$

The approximation $\mathcal{G}$ originates from a data driven, purely mathematical algorithm, which is trained on input-output pairs, yet it neglects the physical constraints and boundaries, whereas $\mathcal{M}$ is used to solve one or multiple differential equations for $\boldsymbol{z}_j^s$. It should be pointed out at this point that in general, $\mathcal{M}$ is not injective, and hence, also not a bijection. That means that the inverse of $\mathcal{M}$ is, strictly speaking, not defined. Our method, however, is aimed at finding a map that is able to provide a fast and accurate point-wise approximation of this relation according to a well-defined accuracy metric. This is in fact similar to

using iterative procedures that aim to approximate the same undefined inverse map.

Furthermore, since $\mathcal{G}$ is solved for $\boldsymbol{\theta}_j^e$,(see Eq. (9)) a final verification step becomes available to approximate the measurement data $\boldsymbol{z}_j^e$ by solving the identified inputs, yielding:

$$\tilde{\boldsymbol{z}}^e \simeq \tilde{\boldsymbol{z}}_{BB}^e = \left\{ \boldsymbol{z}_{j,BB}^s \mid \boldsymbol{z}_{j,BB}^s = \mathcal{M}(\boldsymbol{\theta}_{j,BB}), \boldsymbol{\theta}_{j,BB} = \mathcal{G}(\boldsymbol{z}_j^e), j = 1, \cdots, q \right\} \tag{14}$$

The main drawback of this method, is the prior knowledge required on the input parameters. As the proposed methodology trains $\mathcal{G}$ based upon the available virtual experiments $\mathcal{D}^s$ of n samples, $\mathcal{D}^s$ has to be sampled such that scarce sampled regions are avoided to minimise interpolation error in a possible response surface model. Furthermore, $\mathcal{D}^s$ has be sampled such that the expected identified input parameters are well captured within the limits of the uniformly sampled virtual experiments. This avoids any possible extrapolation using $\mathcal{G}$ on the actual experiments, hence a drawback compared to the optimisation based approach, as the optimisation path does not suffer from this effect.

3 BLACK BOX APPROACH

This section introduces a deep learning method that can be used for the application of inverse uncertainty quantification in process simulation, based on limited experimental data. Normally, inverse uncertainty quantification problems are solved by means of iterative procedures that aim at minimising some discrepancy metric between the measured and simulated responses. The quantification procedure introduced in this paper however starts from the training of a combined unsupervised and supervised learning model $\mathcal{G}$.

3.1 Overview of the approach

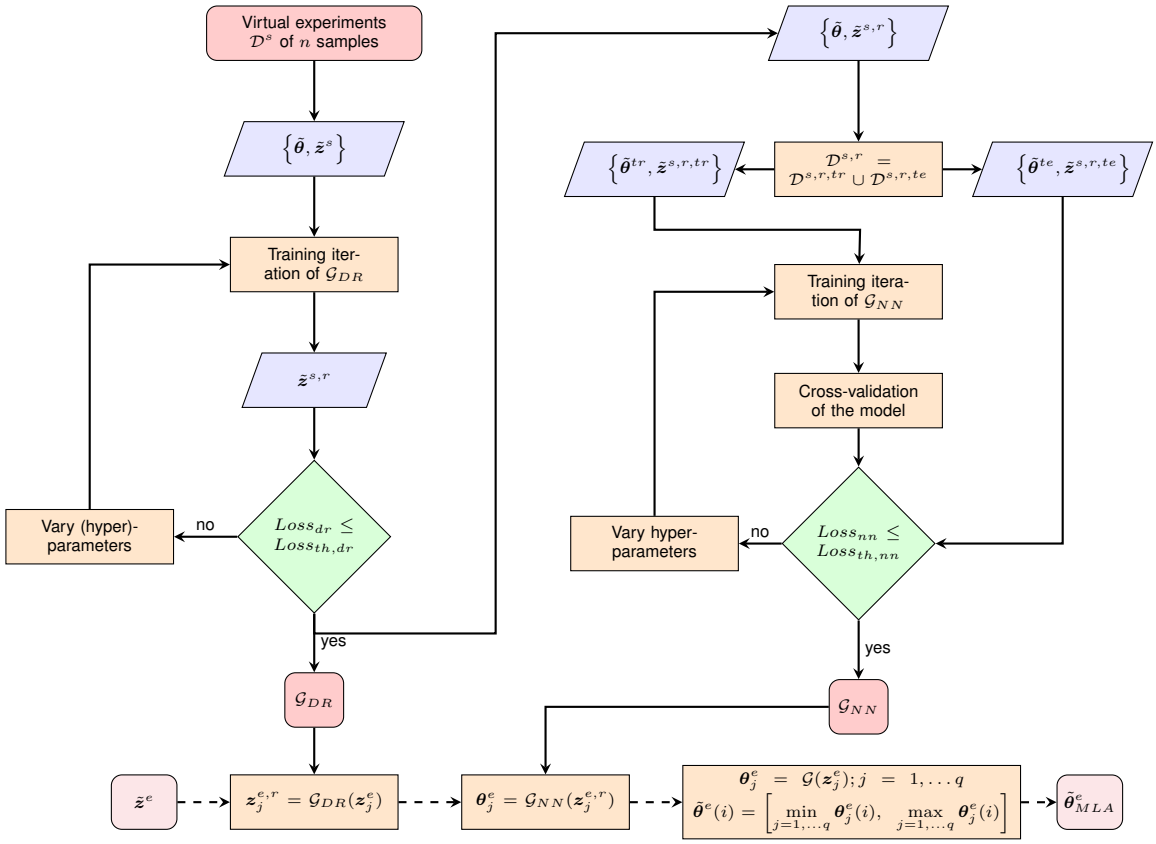


Fig. 1: Workflow, illustrating the proposed prediction structure for inverse UQ

The different steps required to train and test the proposed prediction model are schematically visualised in the flowchart in Figure 1. Two main loops can be distinguished. The first loop aims at defining $\mathcal{G}_{dr}$. The approach starts from a dataset $\mathcal{D}^s$, consisting of a set of model responses $\tilde{\boldsymbol{z}}^s$ and the corresponding

parameters $\tilde{\boldsymbol{\theta}}^s$ and aims at reducing the complexity of the data by applying a dimension reduction technique. There, we compare two commonly used approaches, PCA and autoencoders, to project the set of responses $\tilde{\boldsymbol{z}}^s$ into a reduced space $\tilde{\boldsymbol{z}}^{s,r}$. Depending on the algorithm, the first loop aims at optimising the (hyper-)parameters until a threshold for the loss function $Loss_{th,dr}$ of the dimension reduction is achieved, e.g., by varying the amount of neurons in an autoencoder or the amount of principal components that are neglected in a PCA. When satisfied to the loss condition, a second loop is entered, which first divides the data in a train and test set, which was superfluous until this point, due to the unsupervised approach in the first loop. The second loop aims at defining $\mathcal{G}_{nn}$. A neural network is fitted on a training dataset consisting of input-output pairs, respectively the responses in the reduced space $\tilde{\boldsymbol{z}}^{s,r}$ and the set of model parameters $\tilde{\boldsymbol{\theta}}^s$. A second loss threshold $Loss_{th,nn}$ is introduced to optimise the accuracy of the trained neural network. When satisfied, the architecture and hyper-parameters in $\mathcal{G}_{nn}$ are defined, resulting in a model that is ready to identify model parameters when given experimental data, as introduced in Eq. (9):

$$\boldsymbol{\theta}_j^e = \mathcal{G}(\boldsymbol{z}_j^e) \Longleftrightarrow \boldsymbol{\theta}_j^e = \mathcal{G}_{NN}(\mathcal{G}_{dr}(\boldsymbol{z}_j^e)). \tag{15}$$

The different steps will be described in detail in the following paragraphs.

3.2 Dimension reduction

Experimental data $\tilde{\boldsymbol{z}}^e$ is often largely parameterised, with a corresponding dimension d, where $d \gg 1 \in \mathbb{N}^+$, resulting in a high-dimensional space when representing its volume. The core problem arises from the rapid expansion of this space, leading to sparse and dissimilar data. As a consequence, the computational complexity of a neural network scales exponentially with the data. With each increase in d, connections to every neuron in the subsequent layer are necessitated, according to Eq. (24). This key equation itself is not computationally intensive. However, its recurrence during training, due to the architecture of neural networks, has to be solved numerous times, as such increases the overall computational burden. This phenomenon, commonly termed the curse of dimensionality, pertains to challenges encountered in high-dimensional spaces. It hampers strategies to work efficiently and imposes significant problems concerning computational expenses, which do not occur in low-dimensional spaces [22].

Various techniques for dimensionality reduction are available. They can be divided into convex and non-convex techniques, where convex techniques optimise an objective function that does not contain any local

optima, e.g., Principal Component Analysis (PCA), Kernel PCA, Isomaps, Local Linear Embedding(LLE) and non-convex techniques optimize objective functions that do contain local optima, e.g., Locally Linear Coordination (LLC), manifold charting or autoencoders [23]. It is crucial not only to consider the marginal computation cost that is envisaged for the dimensionality reduction, but also to ensure the capability to embed new high-dimensional data points into an existing low-dimensional data representation. Therefore we select PCA, which is by far the most popular technique, and autoencoders, an efficient, intuitive, non-linear transformation technique, to compare in this work.

3.2.1 PCA

Principal component analysis (PCA) performs the dimension reduction by embedding the data into a linear subspace of lower dimensionality. PCA attempts to find a linear mapping on a basis v that maximises the retained variance in the data by first solving the eigenproblem

$$(S - \lambda I)\, v = 0 \Longleftrightarrow Sv = \lambda v. \tag{16}$$

The eigenproblem is solved for the d principal eigenvalues λ, where $S \in \mathbb{R}^{d \times d}$ is the sample covariance matrix of the data $\tilde{z}^s$, e.g. in the case of 2 input parameters this becomes

$$S = \begin{bmatrix} Var\left[\boldsymbol{z}_1^s\right] & Cov\left[\boldsymbol{z}_2^s, \boldsymbol{z}_1^s\right] \\ Cov\left[\boldsymbol{z}_2^s, \boldsymbol{z}_1^s\right] & \left[\boldsymbol{z}_2^s\right] \end{bmatrix}. \tag{17}$$

The low-dimensional data representations $\tilde{z}^{s,r}$ of the data points $\tilde{z}^s$ are computed by mapping them onto the linear basis v. Since PCA tries to put maximum possible information in the first component, then maximum remaining information in the second and so on, it is an ideal setting for making the data as compact as possible. The dimensionality of the remaining data and the variance that is lost is the only parameter in a PCA model that is user defined. Therefore, the eigenvalues are ranked in descending order, and the

variance that is captured by each eigenvector is determined by

$$\psi_n = \frac{\lambda_n}{\sum_{i=1}^{d} \lambda_i}, \tag{18}$$

where the amount of variance that is lost is given by the remaining overall variance threshold, e.g. $\Psi_{lim} \leq 95\%$, according to

$$\Psi = \sum_{i=1}^{\Psi_{lim}} \psi_i \tag{19}$$

3.2.2 Autoencoder

Autoencoders are opted for their efficiency in terms of model parameters, their capability for non-linear transformations and their intuitive usability. These feed-forward networks are structured with an uneven number of hidden layers L_{hi},with the middle layer containing fewer neurons compared to the first and last layers [24]. This separates the autoencoder in an input layer, an encoder section, a middle layer with $l \leq d$ number of neurons, a decoder section and the reconstructed layer:

$$\tilde{\boldsymbol{z}}_d^s \overset{Encoder}{\rightarrow} \tilde{\boldsymbol{z}}_l^{s,r} \overset{Decoder}{\rightarrow} \tilde{\boldsymbol{z}}_d'^s, \tag{20}$$

where $\tilde{z}_d^s \approx \tilde{z}_d'^s$ by means of a training in an unsupervised manner where its goal is to minimise an error in reconstructing $\tilde{z}_d'^s$ [25]. The input layer $\tilde{z}_d^s$ has d neurons, where each neuron represents an individual parameter from the dataset. This data is reconstructed in the final layer $\tilde{z}_d'^s$ of equal dimension d. The centre layer in the network $\tilde{z}_l^{s,r}$ represents the original data in $\mathbb{R}^l$, with $l < d$ while preserving as much structure as possible from the original data. The resulting number of dimensions l in this centre layer functions as the input for a neural network.

The network is trained using a loss function that minimises the mean square error:

$$Loss_{th,dr}(\tilde{\boldsymbol{z}}_d^s, \tilde{\boldsymbol{z}}_d'^s) = \frac{1}{n} \sum_{i=1}^{n} \left\| \boldsymbol{z}_i^s - \boldsymbol{z}_i'^s \right\|, \tag{21}$$

where the aim is to reduce the disparity between the original $\tilde{\boldsymbol{z}}_d^s$ and reconstructed output $\tilde{\boldsymbol{z}}_d'^s$ across n data points. Estimating the optimal dimension for the middle layer involves determining the intrinsic dimension l_{intr} of the dataset. This intrinsic dimension denotes the minimal number of variables required in the reduced space to represent the full dataset. Regularly used geometric methods exploit the intrinsic geometry of the dataset, typically relying on fractal dimensions or nearest neighbour distances [26]. Among these methods, the correlation dimension is widely utilised. Given a set $S_n = \{\boldsymbol{z}_1^s, \ldots, \boldsymbol{z}_n^s\}$ in a metric space, the correlation dimension is defined as [26]:

$$D_C \equiv \lim_{n\to\infty} \lim_{r\to\infty} \frac{log\ C_m(r)}{log\ r}, \tag{22}$$

with:

$$C_m(r) = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} I\left\{\|\boldsymbol{z}_j - \boldsymbol{z}_i\| \leq r\right\}, \tag{23}$$

where I is the indicator function, r the number of even intervals in the high dimensional hyper-cube and n the number of data points. The correlation dimension is estimated by plotting log $C_n(r)$ against log r and estimating the slope of the linear part of the curve until a predefined cut-off is achieved [26].

3.3 Neural network fitting

The approach proposed in this study aims to circumvent the necessity for extensive computational iterations, which are typically demanded by current state-of-the-art methods. This is achieved by training an Artificial Neural Network (ANN) $\mathcal{G}$ to approximate the interval uncertainty associated with the input parameters $\tilde{\boldsymbol{\theta}}_s$. This is accomplished by feeding the ANN with a set of responses $\tilde{\boldsymbol{z}}^s$ in a reduced space $\tilde{\boldsymbol{z}}^{s,r}$, see Eq. (9). The trained network can than be exploited to find an approximation of the uncertainty in the parameter set $\boldsymbol{\theta}^{\mathbf{I}}$ by feeding it the experimental data in the reduced space $\tilde{\boldsymbol{z}}^{e,r}$.

As illustrated in the flowchart in Figure 1, the model $\mathcal{G}$ is a combination of an unsupervised dimension reduction and a supervised neural network, which yields: $\tilde{\boldsymbol{z}}^s \overset{\mathcal{G}_{DR}}{\rightarrow} \tilde{\boldsymbol{z}}^{s,r} \overset{\mathcal{G}_{NN}}{\rightarrow} \tilde{\boldsymbol{\theta}}^s$. To define $\mathcal{G}_{NN}$, the neural network requires training, wherein the neural network iteratively updates its weights and biases. This process ensures the network's robustness by fine-tuning its parameters to accurately capture the underlying

patterns and relationships within the data. Therefore, the ANN in this work uses the Bayesian regulated back-propagation algorithm.The algorithm works efficiently by computing the gradient slope and simultaneously changing the weight values of the network. This iterative process continues until the overall gradient no longer decreases, indicating optimal adjustment of the network's parameters. The inclusion of Bayesian regularisation integrates Bayes' theorem into the regulation scheme, effectively mitigating discrepancies between training and test data. This integration enhances the network's ability to avoid the common pitfall of over fitting, particularly in the presence of noise problems [27].

The neural network community has adopted various strategies in its quest to identify an optimal network structure for corresponding functions [28]. A network that is too shallow, with only a limited number of neurons, often fails to capture the complexities of non-linear relationships effectively. Conversely, a network that is excessively '*deep*', referring to a network with more than one hidden layer, is prone to encountering significant over fitting issues [27]. Additionally, deep networks can pose computational challenges, as each neuron from each layer l_n is connected with all the neurons from the previous layer l_{n-1}, as defined by:

$$z_{j,l}^{s,r} = \varphi(v_j) = \varphi\left(b + \sum_{i=1}^{k} w_{i,j}.z_{i,l-1}^{s,r}\right). \tag{24}$$

where $z_{j,l}^{s,r}$ is the value z for neuron number j in layer l and ^{s,r} referring to the model responses in the reduced space prior to the network, φ represents the activation function, $z_{i,l-1}^{s,r}$ representing neuron number i from layer $l-1$, $w_{i,j}$ the weight assigned to each connection with the previous layer, k the number of neurons in layer $l-1$ and b the bias added to the summation operator, yielding v_j. In general, it can be observed that an increased depth tends to yield improved generalisation across a diverse range of tasks [29]. However, the selection of the number of layers in neural networks often relies on experimentation, intuition, or the principle of "going for depth."

In this study, a stepped random oriented approach is adopted to determine the optimal network architecture. The training process follows an iterative scheme, aiming to maximise regression between the output and the target across a range of iterations. During each iteration, adjustments are made to the structure and hyper-parameters of the network, e.g., the size of the hidden layer(s), the type of activation function. It is important to highlight that this strategy takes the regression from the test data, rather than the training data, to prevent overfitting. The code is configured to randomise variable parameters within a uniformly distributed region. The data is distributed in training and test data, respectively with a ratio of $95/5\%$. The algorithm

trains a 100 networks before a *step* is allowed. The *step* allows for an extra fully connected, hidden layer in the architecture, to try and limit the size of the network in case this is possible. To prevent over fitting, Bayesian regularisation functions are applied, readily accessible within tools like the deep learning toolbox in *Matlab 2018.b* and newer versions. The loop is iteratively executed until the regression coefficient on the training dataset reaches a minimum threshold of 98%. The duration of training for each network varies, contingent upon factors such as the network's size, dataset dimensions, hardware specifications, and code parallelisation. Consequently, the total training time for each parameter configuration can extend to several hours or more. It's crucial to highlight that the resulting parameters, specifically the number of neurons and layers, may vary in function of the characteristics of the selected dataset. Furthermore, a topic that remains relevant in literature is the required amount of samples to train a network on [30, 31]. There is no 'one size fits all' answer, because the required sample-size depends both on the nature of the problem and the complexity of the architecture implemented [32, 33].

4 CASE STUDY 1:DLR AIRMOD STRUCTURE

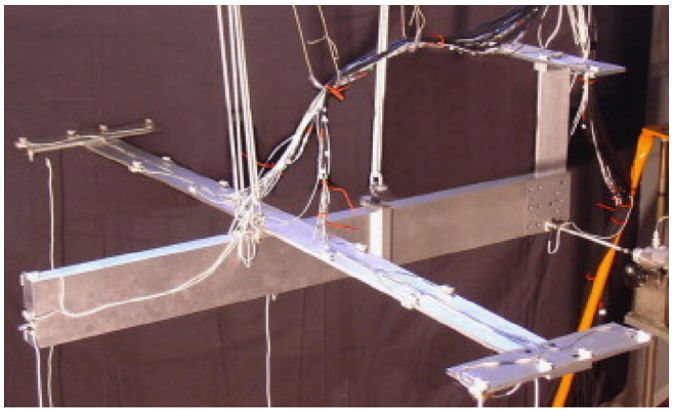


Fig. 2: Illustration of experimental AIRMOD test setup (adapted after [17])

The DLR AIRMOD test structure, designed as a scaled replica of the GARTEUR SM-AG19, serves as a benchmark model for this study. This model, characterized as a 'beam type model' of a generic aeroplane, comprises deterministic properties extracted from axis beam-like components with bolted connections, along with statistics derived from measured eigenfrequencies. These properties collectively represent the main vibrational characteristics of an aircraft. To facilitate stochastic model updating methods, the structure has undergone disassembly and reassembly procedures 130 times, maintaining identical assembly characteristics. A comprehensive set of 18 parameters, encompassing support and joint stiffness values, as well as mass properties, has been carefully selected for analysis. To be consistent with existing literature on the case, certain parameter pairs, specifically the $4^{th} - 5^{th}$ and $9^{th} - 10^{th}$ pairs are considered perfectly dependent. The dataset comprises 86 measured eigenfrequencies, revealing inherent challenges for uncertainty quantification procedures, such as asymmetric modal behavior and closely spaced modes To be consistent with prior studies on the subject, only 14 modes ($1^{st} - 8^{th}, 10^{th} - 12^{th}, 14^{th}, 19^{th}$ and 20^{th}) are selected for identification from the set. These modes encompass suspension motions in yaw, roll, pitch and heave(Modes 1-4), out-of-plane wing bending (Modes 5-8), wing torsion (Mode 10), in-plane wing bending (Modes 11 and 12), and local tail-plane modes (Modes 14, 19, and 20). Automated modal identification was carried out in the work of Govers et al. [17] in which the mode families have been assembled automatically using a pairing algorithm based on the modal assurance criterion (MAC > 0.6) and thresholds for maximum frequency deviations (Df $< 30\%$). For a detailed review of the experimental campaign that was followed to construct this dataset, the reader is referred to [17]. For the purpose of this study, the set of

actual measurements representing z^e are available to validate the methodology and identify the uncertainty in corresponding input parameters $\boldsymbol{\theta}^e$.

4.1 Deep learning

4.1.1 Dimension Reduction

In order to implement the proposed black-box method $\mathcal{G}$ according to Figure 1 , a dimensionality reduction $\mathcal{G}_{dr}$ is required prior to the training of $\mathcal{G}_{nn}$. The pre-selected eigenfrequencies, obtained from the Finite Element Method (FEM) model, serve as input for the dimension reduction. The input data for the FEM model is generated using a Monte Carlo sampling technique, employing a uniform distribution with $\pm 100\%$ bounds on the nominal parameters.

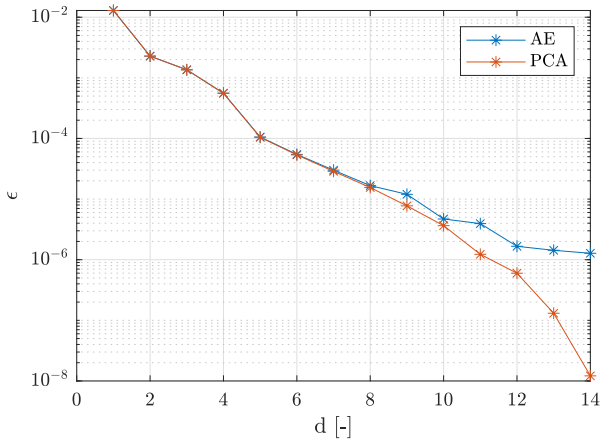


Fig. 3: Reduction error ϵ in function of the reduced dimension for the two compared dimension reduction techniques, based on the results from [34] and [15]

Figure 3 illustrates the reduction error, by means of Mean Squared Error (MSE) in function of the reduced dimension for both a PCA and autoencoder based approach. The mean squared error is computed using Eq. (21). For PCA, ϵ represents the MSE in function of the amount of eigenvectors used for the projection (Eq. (18)). The data is reconstructed by mapping the PCA projections back with the same subset of eigenvectors as the reduced dimension. Alternatively, in case of the autoencoder, ϵ represents the MSE in function of the dimension of the middle layer in the autoencoder. It is noted that for both techniques the MSE decreases when the dimension increases. Furthermore, the difference in MSE from the different approaches is negligible for dimensions lower than 9. It should be noted that, since both algorithms are based on a completely different approach, the almost exact MSE for the lower dimensions is remarkable.

The intrinsic dimension, computed according to Eq. (22) reveals that $d = 11$ for this dataset. Therefore, the set of 14 measured eigenfrequencies is projected trough the encoder part of the autoencoder onto an 11-dimensional space, which serves as the reduced dataset for the remainder in this case. The MSE for encoding the data in a low dimensional space and then decoding it back onto the original space is $4,37.10^{-6}$.

4.1.2 Neural Net Training

After applying the dimension reduction, the second loop from the flowchart in Figure 1 is entered. There, a neural network is trained and optimised to identify the input parameters $\tilde{\boldsymbol{\theta}}^e$. The training dataset is stemming from the responses projected in the reduced space $\boldsymbol{z}^{s,r}$. The data in the reduced space, together with the inputs $\boldsymbol{\theta}^s$ from the FEM model are used as input-output pairs to train the neural network. This training completes the architecture and hyperparameters for $\mathcal{G}_{nn}$, defining the overall model $\mathcal{G}$ (see Eq. (15)). According to the strategy discussed in section 3, a seperate neural network is trained for each input parameter $\theta(i)$. Figure 4 indicates the architectures from each network above the predicted bounds. The architecture can be interpreted as follows: First, the dimension reduction reduces the data from 14 to 11 parameters. Next, to identify parameters $\theta_{1-5}, \theta_{10}$ and θ_{13-16}, only 1 hidden layer is required, while training $\theta_{6-9}, \theta_{11}$ and θ_{12} requires up to 4 hidden layers, respectively with the size of each layer indicated.

4.2 Results

Figure 4 presents the achieved bounds resulting from the proposed deep learning topology method. These bounds, derived by propagating the available measurement data through the trained algorithm $\mathcal{G}$, are represented by interval bars denoted as θ_{BB}. For ease of interpretation, the results are normalized with respect to $\tilde{\theta}_{BB}/\theta_0$. Comparison with the bounds obtained from the interval approach described in [4] is conducted, where the latter is depicted as a blue interval denoted as θ_{MViUQ}. Normalization with an equal normalisation vector as the results from the black box approach $\tilde{\theta}_{MViUQ}/\theta_0$ is applied to the interval approach results, enabling a direct comparison. Minimal discrepancies are observed between the bounds obtained from both methods across most parameters. Parameters θ_7 and θ_{14} demonstrate slight conservatism in each approach, while parameters θ_{11} and θ_{12} demonstrate a slight bias when comparing both approaches. To assess the impact of the obtained offset in the results, the vertexcombinations of the identified bounds are propagated through the FE solver. This verification step yields eigenfrequencies that can be compared with the original experimental data.

Figure 5 illustrates the measured eigenfrequencies for the 86 data points with green dots, using a bee-

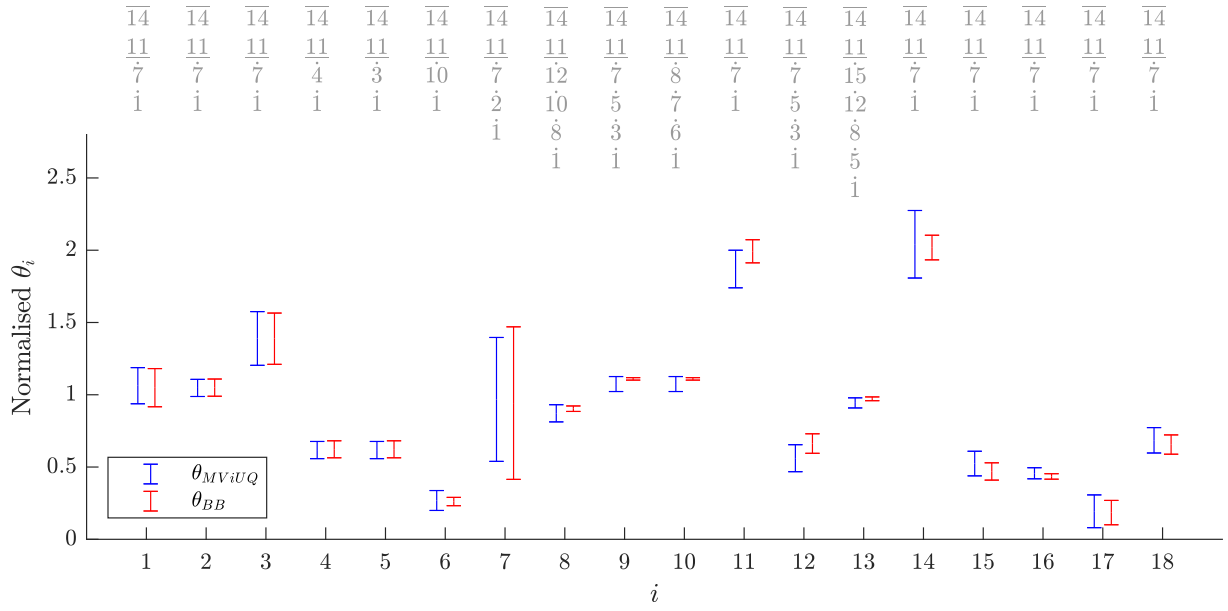


Fig. 4: Normalised bounds of the posterior samples, red bar indicating the interval achieved from the black box model θ_{BB}, compared to the blue visualised intervals achieved via the Multivariate interval quantification θ_{MViUQ} from [4]. The ANN architecture per i is given above the graph, with the dimension reduction part between horizontal lines followed by the size of each hidden layer .

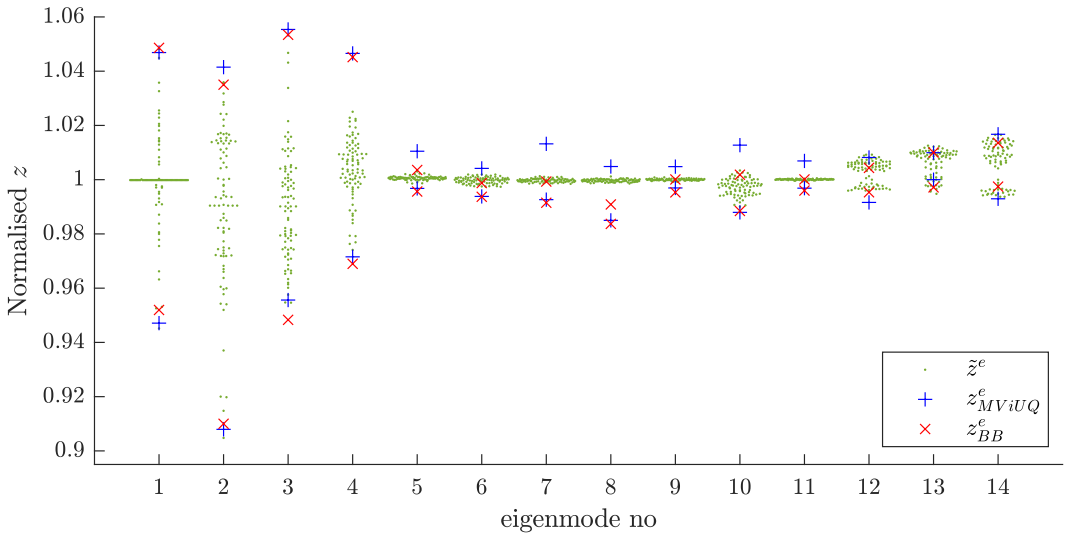


Fig. 5: Normalised measured eigenfrequencies of the AIRMOD structure indicated by the green dots vs. the bounds achieved via propagating the identified input parameters through the forward model, with inputs stemming from: a. The black box model (blue crosses, $\tilde{z}^e_{BB}$); b. The multivariate interval quantification from [4](red crosses,, $\tilde{z}^e_{MViUQ}$).

swarm type plot, with added jitter for improved visualization and to prevent overlap in dense areas. The distribution spread of the measurement data highlights the significance of interval Uncertainty Quantification (iUQ), providing a direct observation of the occurring variability in the identified experimental modal parameters. Notably, distributions can be observed at both lower and higher frequencies, with the plot normalized for improved visualization. The propagated identified bounds from the black box approach are depicted using red crosses. The blue crosses represent the propagated predicted bounds from the multivariate interval approach described in [4]. It is noteworthy that the minimal discrepancies observed in the posterior samples (Fig. 4) correspond with minimal discrepancies when propagated through the FE model. The first four eigenfrequencies, representing the rigid body modes, exhibit a noticeable spread, effectively encapsulated by both approaches with only minor conservatism in the upper bound of the fourth eigenfrequency. Eigenfrequency 5, representing the second wing bending mode, and eigenfrequency 6, representing the third wing bending mode, have minor conservatism in the bounds stemming from the iUQ method. Meanwhile, the proposed black box approach can not completely capture the data, by missing some datapoints with its upper bound. Eigenfrequencies 7 and 8, representing the first asymmetrical and symmetrical wing torsion modes, respectively, are barely or not at all within the bounds, indicating the least accurate results from the black box approach. Similarly, the higher frequencies, including the fourth wing bending mode (eigenfrequency 10), the first and second wing fore-aft bending modes (eigenfrequencies 11 and 12), the first vertical tail piece torsion mode (eigenfrequency 14), the second horizontal tail piece bending mode (eigenfrequency 19), and the first horizontal tail piece fore-aft bending mode (eigenfrequency 20), have their bounds situated at the edges of the data, without entirely encapsulating it.

The computational overhead for processing the measurements through the network is minimal. Training the autoencoder is efficient, with all computations performed using a single thread of an Intel Xeon E5 @ 3.7 GHz processor. Loading the full dataset into memory requires 0.409 seconds, with an additional 23 seconds to achieve optimal training performance for the autoencoder. Despite the longer training time, the autoencoder remains highly employable for real-time applications. Additionally, it is worth noting that due to the unsupervised nature of the autoencoder, labeling of data is unnecessary, rendering it cost-effective. This characteristic allows for denser datasets, which are particularly effective for training purposes. Propagating new data through the autoencoder is efficient, requiring only $0,0284s$ for each measurement. Similarly, testing the network by propagating data vectors from the reduced space to identified parameters only takes $0,0643s$ for the 16 networks corresponding to the 16 parameters.

5 CASE STUDY 2: RESISTANCE PRESSURE WELDING PROCESS

Resistance pressure welding (RPW) is a highly efficient, low cost and easy realisable joining technique. The process employs two welding electrodes that press two or more overlapping sheet-like workpieces together. The heat generated by the then-applied electric current, in correspondence with Joule's law, causes local melting at the workpiece's common faying surface, leading to joining these workpieces. Thanks to its wide-spread use, RPW has grown to a mature joining technique, with literature dating back to the second half of the 20th century (experimentally oriented) [35] and early 2000's (on-line monitoring, Finite Element methods) [36, 37]. In the context of process monitoring, real-time weld quality estimation based on data driven techniques are becoming ever more common [38, 39, 40]. These approaches typically link process parameters and on-line measurements to product quality metrics in order to guard the process. In this respect, deep learning approaches yield very fast black box models enabling on-line application for process control. Yet, in industrial practice, the process still suffers from a high sensitivity to often uncontrollable and variable process conditions, yielding inconsistency of the welds, which generally leads to significant degradation of the weld quality.

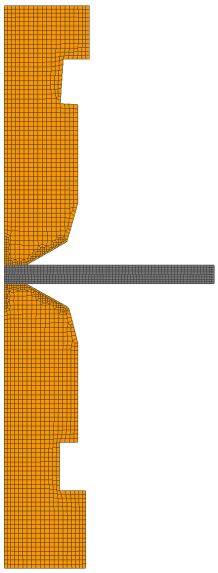


Fig. 6: Cross-section of the axisymmetric RPW model, illustrating two workpieces (individual thickness $\approx 1mm$) that are being welded together (*grey,meshed*) and the two electrodes pressing against the workpieces (*orange,meshed*)

5.1 Overview of the RPW process model

The model in this work is visualised in Figure 6, which represents a simplified 2D-axisymmetric model of a RPW-process, welding with realistic, industry-like boundary conditions and specimen dimensions according to ISO 14270:2016. The nominal process parameters in the model are chosen as such that there is a realistic weld nugget formation according to ISO 10447:2022. Multiple controllable (four) and uncontrollable (two) process conditions are sampled uniformly between decided bounds, based on expert knowledge on the topic. The complete dataset in this work contains 10.000 propagations on a polynomial chaos expansion surrogate model, based on 200 propagations of the FE model, generated based on the code from the UQLab library [41]. The model results in seven measured responses, containing important quality indicators on the process. For this work, the set of input parameters representing $\boldsymbol{\theta}^e$ and the responses representing $\boldsymbol{z}^e$ are available to test the methodology and identify the uncertainty in corresponding input parameters $\tilde{\boldsymbol{\theta}}^e_{BB}$. The model parameters are sampled uniformly, in a latin-hypercube sampling scheme and include respectively, the current $[kA]$, force $[kN]$, initial temperature of the electrode $[^\circ C]$, initial temperature of the workpieces $[^\circ C]$, the film thickness between the electrode and the workpiece $[mm]$ and the film thickness between the workpieces $[mm]$. The responses include the height of the generated weld nugget *(NHS)*, the width of the generated weld nugget *(NWS)*, the heat affected zone *(HAZ)*, the contact area between electrode and workpiece top and bottom *(CWET & CWED)*, the peak temperature in the top workpiece and the bottom workpiece *(MaxT1 & MaxT2)*.

5.2 Deep learning

The first step, prior to training the neural network is a reduction of the dimension of the measured data. The goal in this work is to provide a well performing, yet low-effort approach. For the application of the proposed deep learning method, a reduction of the dimension is applied prior to computing the identified parameters. However, there is no criteria in what or when dimension reduction techniques are beneficial. Therefore, the neural network is trained on both a reduced measurement set $\boldsymbol{z}^{s,r}$, as well as the full dimensional measurement set $\boldsymbol{z}^s$ for a comparison in computational effort and accuracy on this case. Following the notation from the flowchart in Figure 1, $\boldsymbol{z}^s \equiv \boldsymbol{z}^{s,r}$ in the case no technique is used to reduce the dimension of the measurement data. Figure 7 illustrates the reduction error as a function of the reduced dimension. In case of the autoencoder ϵ represents the MSE (Eq. (21)) in function of the dimensionality of the middle layer in the autoencoder and for PCA, ϵ represents the MSE (Eq. (21)) in function of the amount of eigenvectors used for the projection (Eq. (18)). It is noted that for both techniques the MSE decreases

when the dimension increases, while both approaches achieve an almost identical result. It should be noted that, since both algorithms are based on a completely different approach, the negligible difference is remarkable. The application of Eq. (22) to establish the intrinsic dimension of the dataset, reveals $d = 4$.

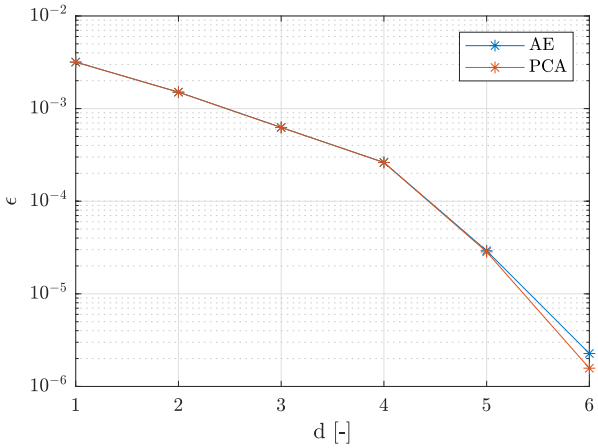


Fig. 7: Reduction error ϵ in function of the reduced dimension for the two compared dimension reduction techniques

Following Eq. (9), the inverse model is trained based on the input-output pairs to determine $\mathcal{G}$, consisting of $\mathcal{G}_{\mathcal{DR}}$ and $\mathcal{G}_{\mathcal{NN}}$. Training the networks according to the method described in section 3 yields networks witch an architecture provided in table 1. To implement the strategy discussed in section 3.3, a neural network is trained for each input parameter $\theta(i)$ separately, with the architecture of each network provided in table 1. The table can be interpreted as follows: to identify parameter θ_1 without a prior dimension reduction, one layer consisting of four neurons is required; with a PCA prior to the neural network, the input dimension is reduced from seven to four, and the neural network has a hidden layer of three neurons and likewise for the case with an autoencoder.

5.3 Results

As proposed in the flowchart in Figure 1, the neural network is trained with the dataset as introduced in 5.1. The thick green line in Figure 8 indicates the bounding corners of the convex hull that covers the uniformly sampled training points for all combinations of θ_i^s. For this case study, the bounds of the sampling are chosen arbitrarily, within the known limits of physically admissible parameters, while keeping a very conservative overestimate on the expected experimental data. For a fair evaluation of the model,

Table 1: Trained ANN architectures, with between brackets the dimension reduction part

θ_i	ANN archit.	PCA-ANN archit.	AE-ANN archit.
1	7 - 4 - 1	(7 - 4) -3- 1	(7 - 4) -3- 1
2	7 - 3 - 1	(7 - 4) -2- 1	(7 - 4) -2- 1
3	7 - 4 - 1	(7 - 4) -3- 1	(7 - 4) -2- 1
4	7 - 6 - 1	(7 - 4) -6- 1	(7 - 4) -6- 1
5	7 - 6 - 1	(7 - 4) -5- 1	(7 - 4) -6- 1
6	7 - 4 - 1	(7 - 4) -3- 1	(7 - 4) -3- 1

the measured responses $\boldsymbol{z}^e$ are based on a new and unseen generated dataset $\mathcal{D}^e$ stemming from the RPW model. To be clear, the data $\mathcal{D}^e$ is not included during the training stage, to provide an unbiased final evaluation of the model fit. This will be referred to as the experimental dataset in further discussion. The experimental set is chosen such that the boundaries are well capturing the expected region of the model, whilst the training set was chosen such that the domain of feasible parameters has a large overlap. This makes sure that in case the model $\mathcal{G}$ has an inaccurate prediction, the model is not extrapolating into an unknown region. The test data are given by the red dots in Figure 8. The identified bounds, established by propagating the available test data through the trained algorithm are given by the blue, purple and red hull, respectively for the pure neural network fit, the neural network fit with prior PCA dimension reduction, and the neural network fit with prior dimension reduction based on an autoencoder. It can be noted that the convex hulls encapsulate the data very well for $\theta_{1,2}$ and $\theta_{5,6}$. However, the model fails in encompassing θ_3 and θ_4 completely and predicts a too narrow interval. In the case where a dimension reduction is applied, the model fails in encompassing θ_4 completely and has a more or less a fixed value corresponding to the centre of the distribution.

As in the previous case, the times required to process the data through the network is marginal. All computations for the training are made using a single-thread of an Intel Xeon E5 3.7 Ghz. To load the full dataset in the memory 0,318 s is required. It takes a total time of 125 s to reach the best training performance of the autoencoder. However this long time is irrelevant regarding real-time applications, as the autoencoder is fully employable to be used as dimension reduction technique. It should be noted that, due to the unsupervised manner of the autoencoder, data should not be labelled and is therefore deemed cheap. This allows for a more dense dataset, which is even more effective for training. Propagating new data through the autoencoder takes only 0,0410 s for a single measurement from the dataset $\boldsymbol{z}^e$. Testing the network by propagating a single data vector from the reduced space $\boldsymbol{z}^{e,r}$ towards the identified parameters

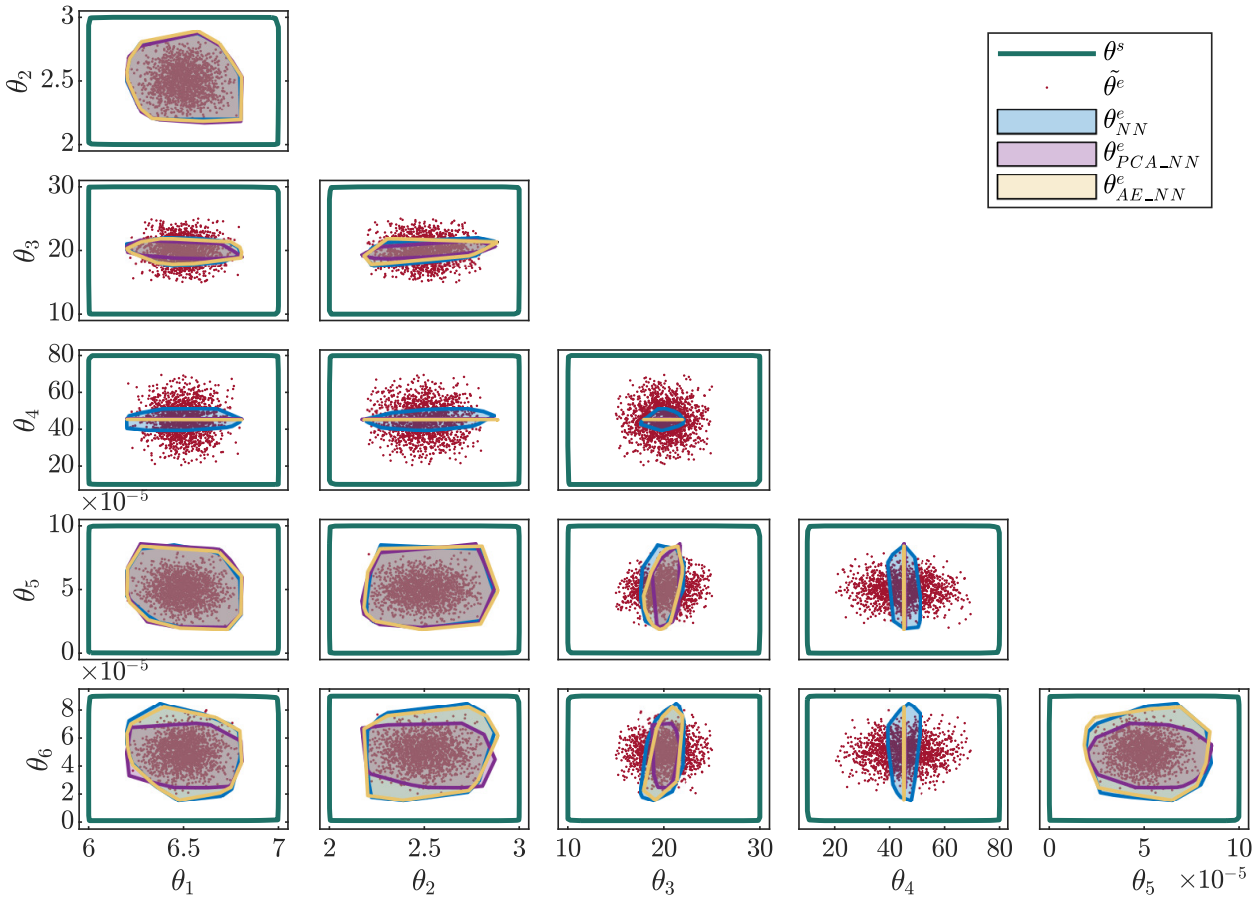


Fig. 8: Scatter plot of all combinations of process parameters. The hull encapsulating the training data is indicated in green. The red dots indicate the test data. The predicted bounds on the test data are indicated for the different proposed techniques in respectively blue, magenta and yellow

$\tilde{\boldsymbol{\theta}}^e$ takes a total of 0,0218 s for the 6 networks, corresponding to the 6 parameters $\theta(i)$.

To verify the accuracy of the obtained identified parameters, the bounds are used as an input for the FE solver (Section 5.1). This results in measurements which can be compared with the experimental data used for the inverse identification. The red dots in Figure 9 represent the experimental data that is used for the inverse identification. The identified bounds of the model parameter ranges from the different approaches are calculated by propagating the vertex-combinations of the identified interval bounds through the FE model and the resulting bounds are illustrated with crosses. It can be concluded that all the experimental data is encapsulated, with a noticeable small conservatism on both the lower and upper bound for the CWED, MaxT1 and MatxT2. Furthermore, it can be concluded that there is less then 1% offset between the different approaches, except for the lower bound of the CWET (2.9%) and the upper bound of the CWED (3.0%).

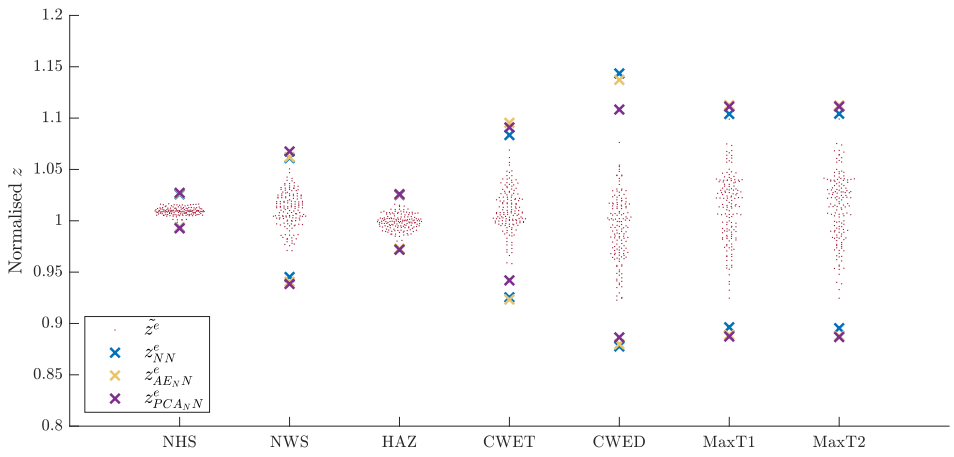


Fig. 9: Responses of the RPW FE-model used for identification are indicated by the blue dots, with upper and lower bound of the responses of the RPW FE-model based on the identified parameters based on different approaches, respectively in black, green and magenta. Note that the centres of the crosses indicates the resulting value.

Further investigation indicates that, for this case, parameter θ_3 and θ_4 have almost no contribution in the variance of the responses from z^e, except for z^e_4 and z^e_5 there is a contribution between 10 and 20%. Therefore it is concluded that the network doesn't necessary fail to establish a proper fit on parameter θ_3 and θ_4, as the complete input-output relation is randomised. The above hypothesis is based on the Sobol indices as provided in Figure 10.

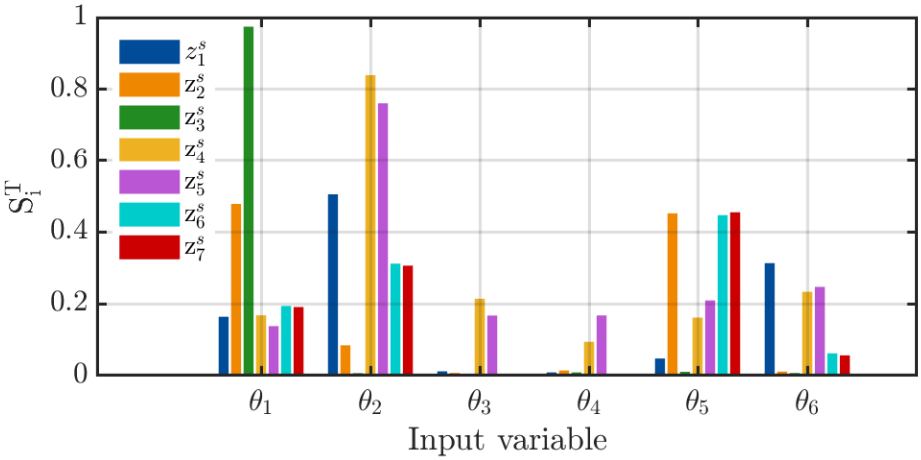


Fig. 10: Total Sobol indices to provide a global sensitivity analysis of the parameters

5.4 Validity of the proposed black-box and its verification

It should be noted that, despite the numerical efficient and accurate results, the proposed approach requires basic knowledge on the imposed problem. On the ground that black boxes are becoming ever more common, one might assume it's plug-and-play. However, as concluded in Section 5.3, and depicted in Fig. 10, the accuracy of the method largely depends on the relevance of the selection of quantities of interest. To demonstrate, we include two deviations on the example.

Significance of process parameters

A first example examines the effect of selecting the correct parameters prone to identification. For this approach, according to the proposed flowchart in Figure. 1, a process parameter is excluded in the training of $\mathcal{G}$. This is done by propagating the FE model n times with non-deterministic parameters, except for one fixed process parameter, which is chosen in this case as the mean nominal value of the dataset. Next, this input-output relation set $\left\{\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{z}}^s\right\}$ is used to train and test the network. Finally, an experimental dataset is generated, where all parameters are sampled within a selected interval, including the parameter that was nominal during the training of the black box. The vertex-combinations of the identified interval bounds are propagated through the FE model. This is repeated for both a parameter with rather high Sobol indices towards responses and made for a parameter with rater low Sobol indices, respectively θ_1 and θ_3 are excluded. The identified bounds in both cases are compared with the set of identified bounds in Section 5. Figure 11 illustrates the results for both cases. For the case where θ_3, an input parameter with low significance level is excluded during training, the resulting identified bounds are almost similar. For most parameters, only minor differences are noticeable. However, θ_4 is now indicating an even larger misfit compared to the original case, see Section. 5. When θ_1 is excluded from training the black box, the results are more noticeable. For both $\theta_{2,3,5,6}$, the identified hull is largely overestimated. θ_4 achieves a similar, but incorrect identification. Fig. 13a further acknowledges this, as the propagated vertex combinations of the identified bounds trough the FE model indicate a similar response. When θ_1 is left out in the training phase, the experimental data is not encapsulated by the bounds anymore, with noticeable offsets for $\boldsymbol{z}^e_{1,2,3,6,7}$ and a large overestimation, exceeding the bounds of the illustration, for $\boldsymbol{z}^e_{4,5}$. When excluding the parameter with low significance level, θ_3, the responses of the propagated parameter vertex combinations trough the FE model deviate slightly, often causing a slight conservatism, but within acceptable limits. This leads to the conclusion that, although the proposed approach is a complete black-box, prior knowledge on the relevance of the data is required for a well-fit model.

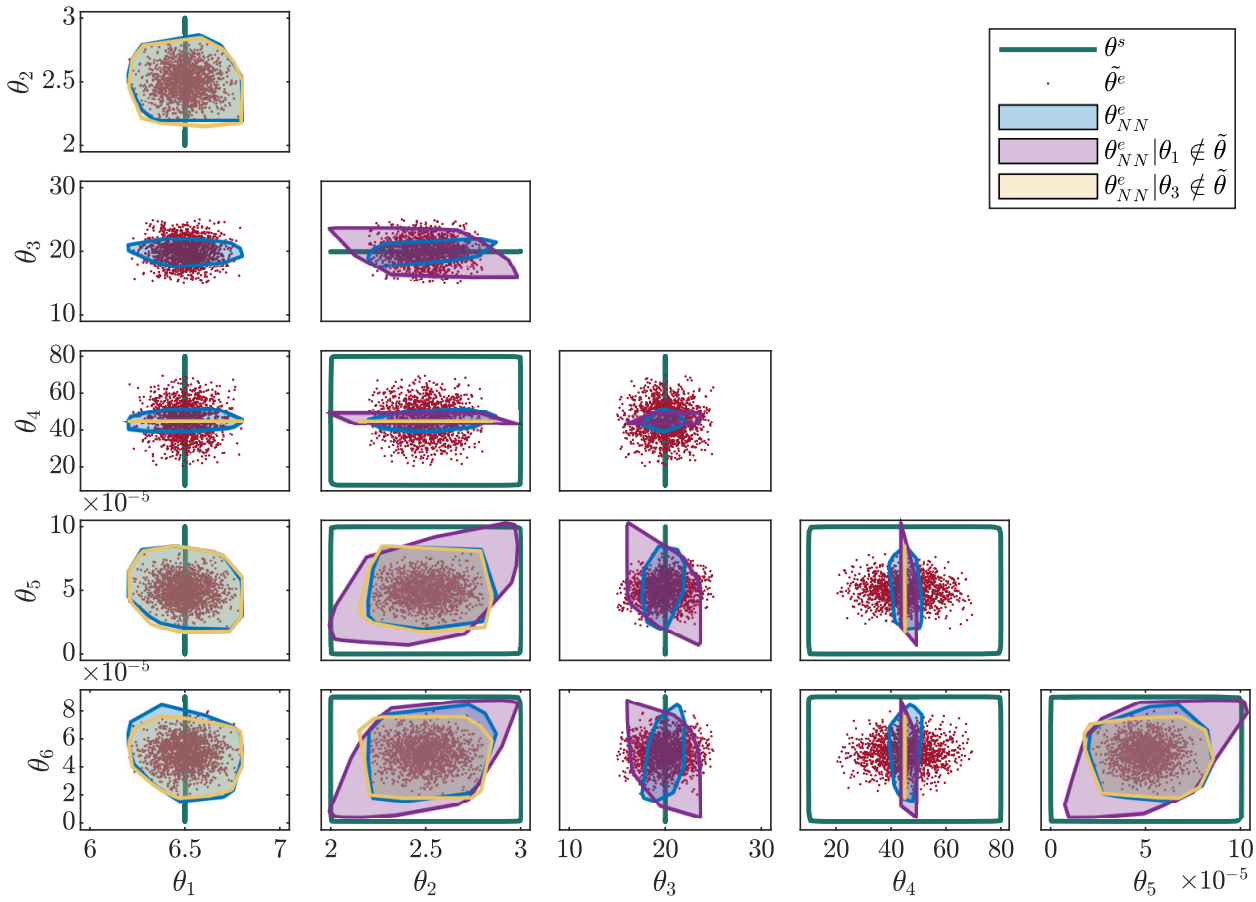


Fig. 11: Identified process parameters; Exclusion of either θ_1 or θ_3 from θ^s

Significance of measurements

A second example examines the effect of missing data, which is often the case when quality indicators are not trivial to measure, or unobservable. This example complies with the proposed flowchart in Figure. 1. The training of $\mathcal{G}$ is executed thrice, consecutively decreasing the amount of information, respectively by leaving $\boldsymbol{z}^e_4$, $\boldsymbol{z}^e_{3,4}$ and $\boldsymbol{z}^e_{2,3,4}$ out of the analysis. After training, when deploying $\mathcal{G}$ for the point-wise identification, the experimental data is also excluding the information on these responses. Figure 12 depicts the results, compared to the trained black-box from Section. 5. It is noticed that, the bounds deteriorate further when less information is available. When excluding all $\boldsymbol{z}^e_{2,3,4}$, the variance in θ_1 is even largely underestimated. Fig. 13b illustrates the propagated vertex combinations of the identified bounds trough the FE model. This indicates minor changes. However, the bounds on $\boldsymbol{z}^e_3$, are not encapsulating the complete experimental dataset when $\boldsymbol{z}^e_3$ or $\boldsymbol{z}^e_{3,4}$ were not part of the training. Reviewing the Sobol indices in Fig. 10 reveals that $\boldsymbol{z}^e_3$ almost solely depends on θ_1, which was the parameter where the extremes in the experi-

mental data were not included in the identified hulls. It can be concluded that measurement data is often coupled, and, therefore, excluding quantities of interest do not per se lead to a worse identification, but indicate that an increase in lack of information lead to a worse identification in an interval context.

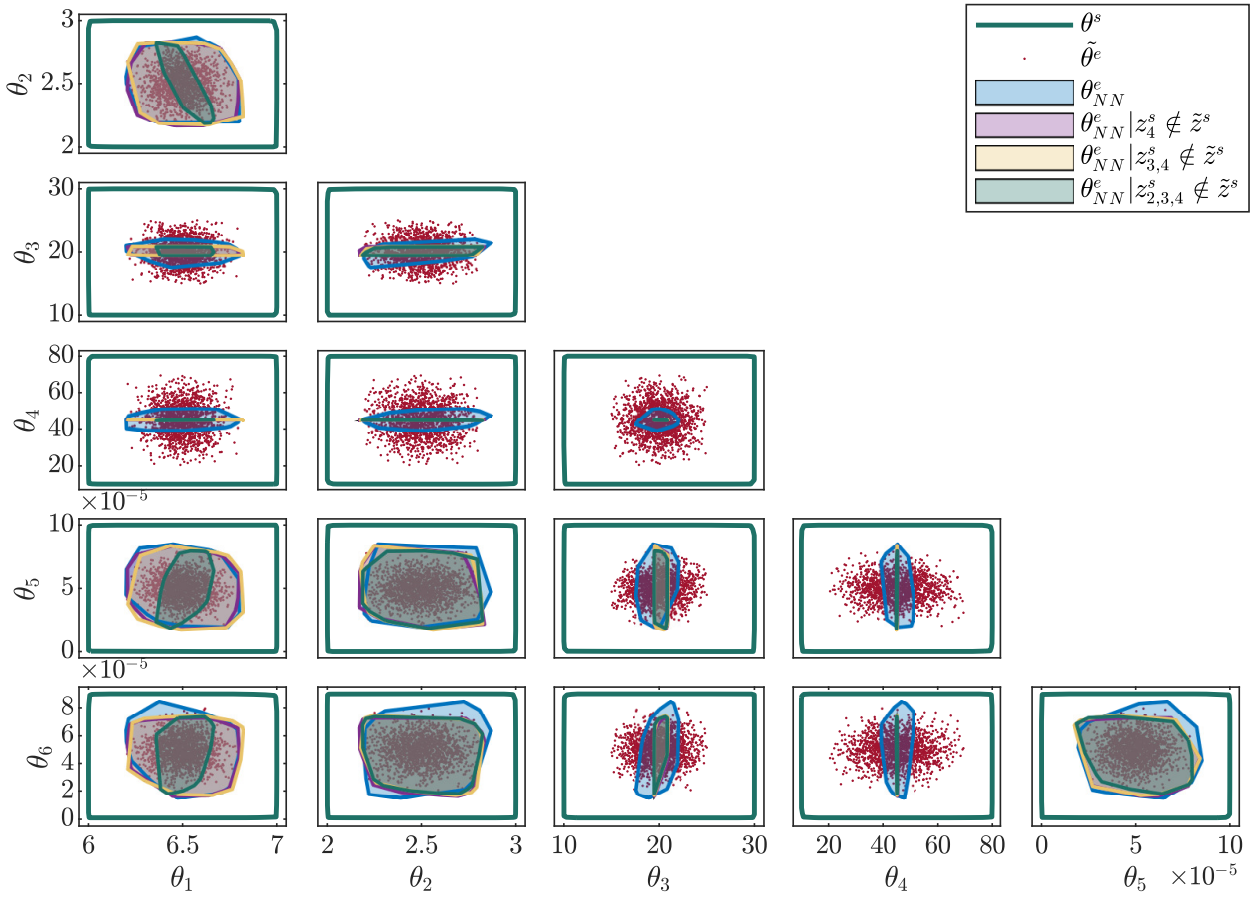
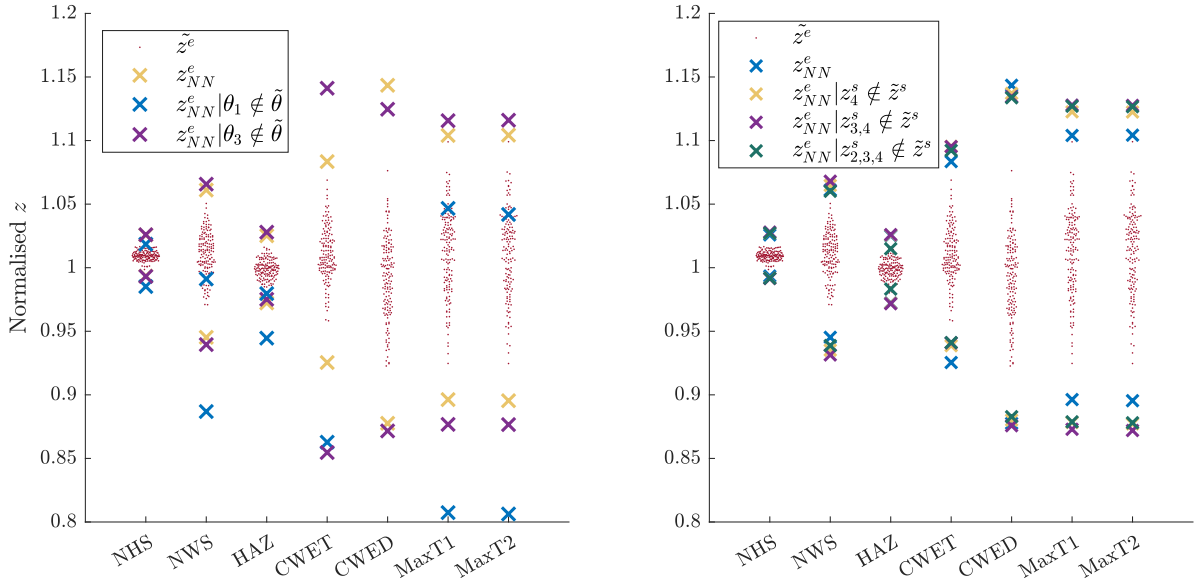


Fig. 12: Identified process parameters; Exclusion of either z^e_4, $z^e_{3,4}$ or $z^e_{2,3,4}$ from $\tilde{z}^s$



(a) Exclusion of either θ_1 or θ_3 from θ^s

(b) Exclusion of either $\boldsymbol{z}^e_4$, $\boldsymbol{z}^e_{3,4}$ or $\boldsymbol{z}^e_{2,3,4}$ from $\tilde{\boldsymbol{z}}^s$

Fig. 13: Results from propagating Identified bounds trough FE model

6 CONCLUSIONS

The aim of this study is to develop an efficient method for the inverse quantification of uncertainty with black box numerical simulation models. Existing methodologies, including probabilistic, interval and imprecise probabilistic techniques, have demonstrated their effectiveness in computing posterior results. However, their iterative nature often demands substantial and unpredictable computation load, rendering them impractible for real-time applications.

The black box approach that is presented in this work integrates a combination of unsupervised and supervised methodologies, resulting in a point-wise identification of input parameters. This technique involves employing dimension-reduction methods such as autoencoders or principal component analysis, prior to an artificial neural network, to identify input parameters based on a set of measurements of the system's responses. By training the methodology on input-output pairs generated by a finite element model of the structure, the necessity for iterative procedures is eliminated. The trained system is capable of processing experimental measurement data to accurately identify input parameters, thereby showcasing its potential for practical application in diverse real-time engineering scenarios.

Two cases are presented in this work, where the first case compares with the existing interval technique and the second case elaborates more on different approaches.

First, to evaluate the efficacy of the proposed methodology, we selected the DLR AIRMOD test structure and its corresponding dataset. This dataset is chosen due to its complexity and the availability of established benchmark solutions. A comparison with the inverse method outlined in [4] demonstrates a notable improvement in numerical efficiency achieved by our black box approach, while maintaining comparable accuracy levels. During the implementation phase, our proposed network training strategy swiftly identifies the specific requirements for each parameter. Some parameters yield minimal error on test data with an ANN featuring only one hidden layer, while others demand deeper networks to achieve satisfactory performance. In terms of accuracy in identified bounds, our proposed black box model delivers results analogue to those obtained from the multivariate interval approach. However, it is worth noting that a few outcomes from the ANN fail to exactly encapsulate the experimental dataset.

The second case is a practical example based on the resistance pressure welding joining technique. The dataset stems from a self developed validated Finite Element model of the process. The identified uncertainty on the model parameters are a very tight fit on four out of six parameters. Further investigation based on global sensitivity analysis indicates that the two inaccurate predicted parameters have a very low influence on the variance in the responses of the model, which is likely the cause for a misfit of the model

on these parameters. Propagating the identified bounds through the Finite Element model yields bounds that encompass the experimental data very well, with only two out of seven outputs for which the predicted range is slightly conservative. Comparing the neural network where there is a prior dimension reduction based on PCA or autoencoders reveals that the offset between the different approaches is limited to 1%, except for the lower bound of the CWET (2.9%) and the upper bound of the CWED (3.0%). Furthermore, the difference in Mean-squared error from both the PCA and autoencoder dimension reduction approach is negligible when reducing the dimensions drastically. It should be noted that, since both algorithms are based on a completely different approach, the almost exact MSE for the lower dimensions is remarkable. Furthermore, the second case is used to provide insight on the importance of process knowledge regarding the context of inverse interval identification. To achieve this, two deviations of the case are depicted, where in each alteration information is left out of the training of the black box. This is obtained by leaving out process parameters or leaving out measurable quantities of interest. Both scenarios contribute to the importance of process knowledge, based on the results of a prior sensitivity analysis on the case study. The results indicate a high loss in accuracy of the obtained identified bounds when a significant parameter is left out, and only a minor, negligible loss in accuracy when an insignificant parameter is left out. Excluding measured quantities of interest in the training indicate a loss in accuracy parallel to the degree of discarded information.

Computations on both cases were executed on commonly available hardware. Training times on both cases were limited, with training times $< 1h$ for both cases, including sampling the predefined surrogate models of the forward model $\mathcal{M}$. Once trained, identifying parameters is extremely fast ($\ll 1s$), from which can be concluded that this approach is feasible in a real-time setting. Limiting even faster identification is only a matter of techniques to load the data in the memory and the efficiency of the all encompassing code. Compared to conventional interval methods, no guarantee of the exact bounds can be proven, however at strongly reduced computational cost, the method has proves its capability to provide reliable estimations of the exact bounds.

ACKNOWLEDGEMENTS

This work was funded by KU Leuven, grant #C24E/21/026.

Conflict of interest: The authors declare no conflict of interest.

REFERENCES

[1] Stefanou, G., 2009, "The stochastic finite element method: past, present and future," *Computer methods in applied mechanics and engineering,* **198**(9-12), pp. 1031–1051.

[2] Faes, M., and Moens, D., 2020, "Recent trends in the modeling and quantification of non-probabilistic uncertainty," *Archives of Computational Methods in Engineering,* **27**, pp. 633–671.

[3] Faes, M. G., Daub, M., Marelli, S., Patelli, E., and Beer, M., 2021, "Engineering analysis with probability boxes: A review on computational methods," *Structural Safety,* **93**, p. 102092.

[4] Faes, M., Broggi, M., Patelli, E., Govers, Y., Mottershead, J., Beer, M., and Moens, D., 2019, "A multivariate interval approach for inverse uncertainty quantification with limited experimental data," *Mechanical Systems and Signal Processing,* **118**(March), pp. 534–548.

[5] Broggi, M., Faes, M., Patelli, E., Govers, Y., Moens, D., and Beer, M., 2018, "Comparison of Bayesian and interval uncertainty quantification: Application to the AIRMOD test structure," In 2017 IEEE Symposium Series on Computational Intelligence, SSCI 2017 - Proceedings, pp. 1–8.

[6] Beer, M., Ferson, S., and Kreinovich, V., 2013, "Imprecise probabilities in engineering analyses," *Mechanical Systems and Signal Processing,* **37**(May-June), pp. 4–29.

[7] Bi, S., He, K., Zhao, Y., Moens, D., Beer, M., and Zhang, J., 2022, "Towards the nasa uq challenge 2019: Systematically forward and inverse approaches for uncertainty propagation and quantification," *Mechanical Systems and Signal Processing,* **165**, p. 108387.

[8] Zhao, Y., Yang, J., Faes, M. G., Bi, S., and Wang, Y., 2022, "The sub-interval similarity: A general uncertainty quantification metric for both stochastic and interval model updating," *Mechanical Systems and Signal Processing,* **178**, p. 109319.

[9] Sudret, B., 2008, "Global sensitivity analysis using polynomial chaos expansions," *Reliability engineering & system safety,* **93**(7), pp. 964–979.

[10] Bach, F., 2017, "Breaking the curse of dimensionality with Convex Neural Networks," *Journal of Machine Learning Research,* **18**(April), pp. 1–53.

[11] Mak, K. K., Kounseok, L., and Cheolyong, P., 2019, "Applications of machine learning in addiction studies: A systematic review," *Psychiatry Research,* **275**(May), pp. 53–60.

[12] Stetco, A., Nenadic, G., Flynn, D., Barnes, M., Zhao, X., Dinmohammadi, F., Keane, J., and Robu, V., 2019, "Machine learning methods for wind turbine condition monitoring: A review," *Renewable Energy,* **133**(April), pp. 620–635.

[13] Olshausen, B. A., and Field, D. J., 1997, "Sparse coding with an overcomplete basis set: A strategy

employed by v1?," *Vision research,* **37**(23), pp. 3311–3325.

[14] Namazi, M., Karimi-Jafari, M. H., Qassemi, F., and Ghasemi, J. B., 2023, "Autoencoders in generative modeling, feature extraction, regression, and classification," In *Machine Learning and Pattern Recognition Methods in Chemistry from Multivariate and Data Driven Modeling*. Elsevier, pp. 119–136.

[15] Faes, M., Cerneels, J., Vandepitte, D., and Moens, D., 2017, "Identification and quantification of multivariate interval uncertainty in finite element models," *Computer Methods in Applied Mechanics and Engineering,* **315**(March), pp. 896–920.

[16] Bogaerts, L., Faes, M., and Moens, D., 2019, "A fast inverse approach for the quantification of set-theoretical uncertainty," In 2019 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, pp. 768–775.

[17] Govers, Y., Haddad Khodaparast, H., Link, M., and Mottershead, J. E., 2014, "Stochastic Model Updating of the DLR AIRMOD Structure," In Second International conference on vulnerability and risk analysis and management (ICVRAM) and the sixth international symposium on uncertainty modeling and analysis (ISUMA), no. August in ICVRAM proceedings, pp. 475–484.

[18] Callens, R. R., Faes, M. G., and Moens, D., 2021, "Local explicit interval fields for non-stationary uncertainty modelling in finite element models," *Computer Methods in Applied Mechanics and Engineering,* **379**, p. 113735.

[19] Dang, C., Wei, P., Faes, M. G., Valdebenito, M. A., and Beer, M., 2022, "Interval uncertainty propagation by a parallel bayesian global optimization method," *Applied Mathematical Modelling,* **108**, Aug., p. 220–235.

[20] Hanss, M., 2002, "The transformation method for the simulation and analysis of systems with uncertain parameters," *Fuzzy Sets and systems,* **130**(3), pp. 277–289.

[21] Faes, M., and Moens, D., 2019, *Recent Trends in the Modeling and Quantification of Non-probabilistic Uncertainty* No. 0123456789. Springer Netherlands.

[22] Fernández-Martínez, J. L., and Fernández-Muñiz, Z., 2020, "The curse of dimensionality in inverse problems," *Journal of Computational and Applied Mathematics,* **369**, p. 112571.

[23] Hinton, G. E., and Salakhutdinov, R., 2006, "Reducing the Dimensionality of Data with Neural Networks," *Science,* **313**(July), pp. 504–507.

[24] Møller, M. F., 1993, "Efficient Training of Feed-Forward Neural Networks," PhD thesis, Aarhus University.

[25] van der Maaten, L., Postma, E., and van den Herik, J., 2009, "Dimensionality Reduction: A Compara-

tive Review," *J Mach Learn Res,* **10**(October), pp. 1–35.

[26] Strange, H., and Zwiggelaar, R., 2014, *Open Problems in Spectral Dimensionality Reduction* Springer.

[27] Oparaji, U., Sheu, R. J., Bankhead, M., Austin, J., and Patelli, E., 2017, "Robust artificial neural network for reliability and sensitivity analyses of complex non-linear systems," *Neural Networks,* **96**(December), pp. 80–90.

[28] Sadeghi, J., de Angelis, M., and Patelli, E., 2019, "Efficient Training of Deep Neural Networks for Dealing with Incertitude and Big Training Data," *Neural Networks,* **118**(October), pp. 338–351.

[29] Goodfellow, I., Bengio, Y., and Courville, A., 2016, *Deep Learning* MIT Press.

[30] Nasir, V., and Sassani, F., 2021, "A review on deep learning in machining and tool monitoring: methods, opportunities, and challenges," *The International Journal of Advanced Manufacturing Technology,* **115**(9-10), pp. 2683–2709.

[31] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., and Farhan, L., 2021, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data,* **8**, pp. 1–74.

[32] Sarker, I. H., 2021, "Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions," *SN Computer Science,* **2**(6), p. 420.

[33] Mehrish, A., Majumder, N., Bharadwaj, R., Mihalcea, R., and Poria, S., 2023, "A review of deep learning techniques for speech processing," *Information Fusion*, p. 101869.

[34] Bogaerts, L., Faes, M., and Moens, D., 2019, "A Machine Learning Approach For the Inverse Quantification of Set-Theoretical Uncertainty," In 3rd International Conference on Uncertainty Quantification in Computational Sciences and Engineering, pp. 1–13.

[35] Dickinson, D. W., Franklin, J. E., and Stanya, A., 1980, "Characterization of Spot Welding Behavior By Dynamic Electrical Parameter Monitoring.," *Welding Journal (Miami, Fla),* **59**(6).

[36] Hao, M., Osman, K. A., Boomer, D. R., and Newton, C. J., 1996, "Developments in characterization of resistance spot welding of aluminum," *Welding Journal (Miami, Fla),* **75**(1), pp. 1–s.

[37] Hamedi, M., and Atashparva, M., 2017, "A review of electrical contact resistance modeling in resistance spot welding," *Welding in the World,* **61**(2), pp. 269–290.

[38] Brunton, S. L., and Kutz, J. N., 2019, *Data-driven science and engineering: Machine learning, dynamical systems, and control* Cambridge University Press.

[39] Wan, X., Wang, Y., Zhao, D., Huang, Y. A., and Yin, Z., 2017, "Weld quality monitoring research in small scale resistance spot welding by dynamic resistance and neural network," *Measurement: Journal*

of the International Measurement Confederation, **99**, pp. 120–127.

[40] Bogaerts, L., Dejans, A., Faes, M. G., and Moens, D., 2023, "A machine learning approach for efficient and robust resistance spot welding monitoring," *Welding in the World,* **67**(8), pp. 1923–1935.

[41] Marelli, S., and Sudret, B., 2014, "Uqlab: A framework for uncertainty quantification in matlab," In *Vulnerability, uncertainty, and risk: quantification, mitigation, and management*. pp. 2554–2563.