Passive safety systems analysis: A novel approach for inverse uncertainty quantification based on Stacked Sparse Autoencoders and Kriging metamodeling

Passive safety systems analysis: a novel approach for Inverse Uncertainty Quantification based on Stacked Sparse Autoencoders and Kriging Metamodeling

Giovanni Roma[1], Federico Antonello[1], Francesco Di Maio[1*], Nicola Pedroni[2], Enrico Zio[1,4], Andrea Bersano[3], Cristina Bertani[2], Fulvio Mascari[3]

[1] Energy Department, Politecnico di Milano, Via La Masa 34, Milano, 20156, Italy.

[2]Energy Department Politecnico di Torino Corso Duca degli Abruzzi, 24, Torino, 10129, Italy

[3]ENEA- BOLOGNA, Via Martiri Monte Sole 4, Bologna, 40129, Italy

[4]MINES ParisTech, PSL Research University, CRC, Sophia Antipolis, France

*Corresponding Author: francesco.dimaio@polimi.it

Abstract

In passive safety system analysis, it is important to provide the uncertainty quantification of the Thermal-Hydraulic (T-H) code output (e.g., the amount of energy exchanged by the passive safety system during an accidental transient). This requires setting proper Probability Density Functions (PDFs) to represent the uncertainty of selected code inputs and the propagation of this uncertainty through the code. One way to obtain the PDF is by Inverse Uncertainty Quantification (IUQ) methods, which rely directly on experimental data and code simulation results. In this work, we propose a novel IUQ approach based on: (i) Stacked Sparse Autoencoders (SSAEs) to reduce the problem dimensionality; and (ii) Kriging metamodels to decrease the computational cost associated with the posterior sampling of the uncertain input parameters by Markov Chain Monte Carlo (MCMC) (for which many model simulations are typically required). The novelty, in particular, stands in the use of SSAEs for dimensionality reduction: this allows using directly the raw data available from experimental facilities or computer codes (typically characterized by small signal-to-noise ratios), without having to resort to filtering techniques, whose choice and setting are nontrivial and bias the results. The proposed approach is applied to the power exchanged by the Heat Exchanger (HX) predicted by the RELAP5-3D model of the PERSEO (In-Pool Energy Removal System for Emergency Operation) facility, located at the SIET laboratory (Piacenza, Italy), for the test 7 part 2 and characterized by a small signal-to-noise ratio (SNR) value. A comparison between Principal Component Analysis (PCA) and SSAE is carried out to provide some insights on the technical issues associated with the implementation and the use of the approach in the context of IUQ and to highlight the main advantages and drawbacks while also showing the suitability to deal with non-filtered (raw) data.

Keywords: Passive safety systems, Inverse uncertainty quantification, Bayesian inference, Kriging surrogate modeling, Autoencoders.

Acronyms

ANN: Artificial Neural Network
BCs: Boundary conditions
BE: Best Estimate
BEPU: Best Estimate Plus Uncertainty
CV: Cross Validation
DAE: Denosing Autoencoder
DNN: Deep Neural Network
DOE : Design Of Experiment
EM: Expectation-Maximization
ENEA: Agenzia nazionale per le nuove tecnologie, l'energia e lo sviluppo economico sostenibile
HX: Heat Exchanger
HXP: Heat Exchanger Pool
ICs: Initial Conditions
IUQ: Inverse Uncertainty Quantification
KDE: Kernel Density Estimation
LHS : Latin Hypercube Sampling
LOOCV: Leave-One-Out Cross Validation
MAP: Maximum A Posteriori
MCMC: Markov Chain Monte Carlo
MLE: Maximum Likelihood Estimation
MSE: Mean Squared Error
NPPs: Nuclear Power Plants
OP: Overall Pool
PCA: Principal Component Analysis
PCs: Principal Component Vectors
PDF: Probability Density Function
PERSEO: in-Pool Energy Removal System for Emergency Operation
PV: Pressure Vessel
SAE: Sparse Autoencoder
SSAE: Stacked Sparse Autoencoder
SNR: Signal to Noise Ratio
T-H: Thermal-Hydraulic
TV: Triggering Valve
VAE: Variational Autoencoder

List of symbols

Symbols	Dimension	Description
$\boldsymbol{d}$	$\boldsymbol{1 \times 1}$	Number of calibration parameters
$\boldsymbol{q}$	$\boldsymbol{1 \times 1}$	Number of design variables
$\boldsymbol{m}$	$\boldsymbol{1 \times 1}$	Number of design points
$\boldsymbol{p}$	$\boldsymbol{1 \times 1}$	Model output dimension
$\boldsymbol{N_{exp}}$	$\boldsymbol{1 \times 1}$	Number of independent experimental measurements
$\boldsymbol{\theta}$	$\boldsymbol{d \times 1}$	Calibration parameters vector
$\boldsymbol{x}$	$\boldsymbol{q \times 1}$	Design variable vector
$\boldsymbol{y^M(\theta)}$	$\boldsymbol{p \times 1}$	RELAP5-3D computer model output for Test 7 pt.2
$\boldsymbol{y^E}$	$\boldsymbol{p \times 1}$	Experimental data
$\boldsymbol{\check{y}^E}$	$\boldsymbol{p \times 1}$	Reconstructed experimental data
$\boldsymbol{\epsilon}$	$\boldsymbol{p \times 1}$	Measurement error
$\boldsymbol{I\sigma^2_{exp}}$	$\boldsymbol{p \times p}$	Covariance matrix for the measurement error
$\boldsymbol{\Sigma_{exp}}$	$\boldsymbol{p^* \times p^*}$	Measurement error covariance matrix in the $\boldsymbol{p}^*$-dimensional reduced space
$\boldsymbol{\Sigma_{MM}}$	$\boldsymbol{p \times p}$	Covariance matrix for the code uncertainty
$\boldsymbol{\Sigma_{Kriging}}$	$\boldsymbol{p^* \times p^*}$	Covariance matrix for the code uncertainty in the reduced space
$\boldsymbol{\Theta}$	$\boldsymbol{m \times d}$	Ensemble of $\boldsymbol{m}$ design points
$\boldsymbol{Y}$	$\boldsymbol{p \times m}$	Ensemble of $\boldsymbol{m}$ BE computer model outputs
$\boldsymbol{\overline{\mu}_Y}$	$\boldsymbol{p \times 1}$	Column vector of the row means of $\boldsymbol{Y}$
$\boldsymbol{p^*}$	$\boldsymbol{1 \times 1}$	Dimension of the features subspace
$\boldsymbol{\Phi}$	$\boldsymbol{p^* \times p}$	Trasformation matrix (PCA)
$\boldsymbol{\hat{z}^{MM}}$	$\boldsymbol{p^* \times 1}$	Kriging metamodel prediction in the features subspace
$\boldsymbol{z^{MM}}$	$\boldsymbol{p^* \times 1}$	Distribution of the Kriging metamodel prediction in the features subspace
$\boldsymbol{z^E}$	$\boldsymbol{p^* \times 1}$	Experimental data projected in the feature subspace
$\boldsymbol{Z}$	$\boldsymbol{p^* \times m}$	Ensemble of $\boldsymbol{m}$ RELAP5-3D simulation outputs transformed into the feature subspace
$\boldsymbol{z_k}$	$\boldsymbol{1 \times 1}$	$\boldsymbol{k^{th}}$ entry of a vector transformed into the feature subspace
$\boldsymbol{L}$	$\boldsymbol{1 \times 1}$	Number of the SSAE hidden layers
$\boldsymbol{K_l}$	$\boldsymbol{1 \times 1}$	Dimension of the $\boldsymbol{l^{th}}$ SSAE hidden layer
$\boldsymbol{z_{(l)}}$	$\boldsymbol{K_l \times 1}$	Neurons output of the $\boldsymbol{l^{th}}$ hidden layer[1]
$\boldsymbol{\sigma(\cdot)}$	$-$	Sigmoid activation function
$\boldsymbol{W_{(l)}}$	$\boldsymbol{K_l \times K_{l-1}}$	Weight matrix of the $\boldsymbol{l^{th}}$ SSAE layer
$\boldsymbol{b_{(l)}}$	$\boldsymbol{K_l \times 1}$	Bias vector of the $\boldsymbol{l^{th}}$ SSAE layer
$\boldsymbol{E}$	$\boldsymbol{1 \times 1}$	SSAE cost function
$\boldsymbol{R_{error}}$	$\boldsymbol{1 \times 1}$	Reconstruction error (adopted for both the SSAE and the PCA)
$\boldsymbol{R_{sparse}}$	$\boldsymbol{1 \times 1}$	Sparsity regularization term
$\boldsymbol{\beta}$	$\boldsymbol{1 \times 1}$	Coefficient for $\boldsymbol{R_{sparse}}$
$\boldsymbol{R_{L2}}$	$\boldsymbol{1 \times 1}$	L2 regularization term
$\boldsymbol{\lambda}$	$\boldsymbol{1 \times 1}$	Coefficient for $\boldsymbol{R_{L2}}$
$\boldsymbol{\rho}$	$\boldsymbol{1 \times 1}$	Desired averaged neuron activation
$\boldsymbol{N}$	$\boldsymbol{1 \times 1}$	Number of RELAP5-3D simulations adopted for the Forward Uncertainty Propagation

[1] When the subscript is within brackets, $\boldsymbol{z_{(l)}}$ indicates the l^{th} hidden layer output of the SSAE (i.e., a vectorial quantity); otherwise, z_k indicates the k^{th} entry of a vector transformed into the p^*- dimensional feature subspace.

1 Introduction

Over the last decades, nuclear safety analysis frameworks based on Best Estimate Plus Uncertainty (BEPU) approaches for thermal-hydraulics transient calculations have gathered great interest [1,2]. These frameworks stand on Best Estimate (BE) Thermal-Hydraulic (T-H) codes to compute the safety margins of relevant parameters (e.g., fuel pellet maximum centerline temperature) during Nuclear Power Plants (NPPs) accidental scenarios [2]. The computation of the safety margins, taking into account the uncertainty of the calculation, requires identifying the main sources of uncertainty, selecting the most relevant uncertain input parameters and propagating the input uncertainties through the BE T-H code. One of the daunting issues related to this is quantifying the epistemic uncertainty that affects some of the input parameters, i.e., the uncertainty derived from incomplete information on the phenomena, which these parameters describe [3–7]. Traditionally, probability constitutes the mathematical structure used to represent epistemic uncertainty and expert judgment is typically used to specify Probability Density Functions (PDFs), nominal values and upper and lower bounds of the parameters [2,8,9]. Such ad-hoc expert judgment can be aided by Inverse Uncertainty Quantification (IUQ), a powerful tool that determines the uncertainty of the parameters relying on available experimental data and code simulation results [10].

Different approaches have been developed to carry out IUQ. Among them, Bayesian inference is considered the standard approach [11,12], wherein the computation of the posterior PDF is typically tackled through Markov Chain Monte Carlo (MCMC) sampling [13]. MCMC is a class of algorithms that allows sampling from a PDF known up to a normalization constant by running numerous code simulations (up to hundreds of thousands). However, in the context of NPP safety analysis, BE computer models (e.g. RELAP, etc.) are computationally expensive for the implementation of MCMC sampling that becomes practically unfeasible with the current computational resources available. As an example, MCMC sampling with 10^5 iterations with a BE model that takes 1 hour per run would last more than 11 years. To tackle this computational issue, having as a target the prediction of specific phenomena and the related parameters, the BE computer model can be replaced by a computationally cheaper metamodel [14], which is a functional approximation of the input/output relations underlying the original BE code [15,16]. For a comprehensive survey of inverse uncertainty quantification methods applied to nuclear system thermal-hydraulics problems, refer to [17]. Among the various metamodeling approaches [16], Kriging, which assumes that the model output is a realization of a Gaussian Process [18,19], has been successful in various IUQ problems [14,20–24]. The advantage of Kriging is that it provides a measure of the uncertainty associated with each prediction [19].

When the BE model output is a time-dependent scalar quantity (i.e., a time series), at least four different approaches can be adopted to build a metamodel. The first one considers the time-dependent scalar output as a vectorial quantity and builds independent metamodels for each time instance; nevertheless, this approach may result in a significant loss of information, since the multiple outputs can be highly correlated

[25]. Another method is to treat time as an additional input [14]; however, in this approach the number of training points becomes dramatically high in very long time series. For example, if the BE model outputs the quantity of interest at 1000 time instances, with a Design Of Experiment (DOE) of 100 sample points for other inputs such as calibration parameters and design variables, the total number of training points for the Kriging metamodel would be $1000 \cdot 100 = 100000$. The third alternative approach is to use a multi-output emulator (e.g., a Kriging metamodel for predicting a p-dimensional vectorial quantity, where p is the number of time instances) [25,26]; this approach outperforms emulators with time as an additional input [27] but, for very large p, even multi-output emulators may experience a reduction in the metamodeling efficiency [28]. The fourth alternative approach has been developed to address the abovementioned limitations for high-dimensional time series and consists in performing a dimensionality reduction to retain a relatively small number of significant features to represent the entire output space. In fact, when the output data is redundant (e.g., the BE computer model responses for nearby time instances are strongly correlated) and its dimensionality p is too large to be processed (e.g., through a single multi-output emulator), a reduced set of p^* features containing the most relevant information of the original data can be used instead of the whole set of data and a separate metamodel can be built for each of the p^* extracted features.

Principal Component Analysis (PCA) is one of the most common approaches to perform dimensionality reduction [29]. PCA is a linear transformation from a high dimensional space onto a lower-dimensional space, such that the transformed variables are uncorrelated and retain as much as possible of the variance of the original data set [30]. Higdon et al. [31] proposed a method for the representation of time series outputs based on PCA for the dimensionality reduction and Kriging metamodels for the emulation of each Principal Component (PC). PCA has been used in different Bayesian IUQ/calibration problems with high-dimensional outputs for constructing fast-running metamodels [20,31–34]. However, linear dimensionality reduction techniques like PCA cannot deal with complex (real-world) non-linear data [29]. Furthermore, for the specific case of interest here, BE code results may be affected by higher noise (maybe due to numerics or correlation errors [35]) in comparison with the experimental data in relation to the power exchanged by the HX [47] and typically require pre-processing by the analyst (e.g., filtering of the available raw data) [35], which may add a bias. To overcome the limitations of PCA when dealing with time series with small signal-to-noise ratio values, in this work, we explore the use of Autoencoders (AEs) for dimensionality reduction.

An AE is a type of Artificial Neural Network (ANN) designed to learn a new representation of the data by reconstructing the input data itself [36]. It is composed by an encoder and a decoder network. The encoder transforms the high-dimensional data into a small number of features (i.e., into a lower-dimensional representation), whereas the decoder recovers the high-dimensional data from the features. AEs are widely used to perform dimensionality reduction [37–39], machine health monitoring [36], and image processing [40]. Over the last years, several variants of AEs have been developed, such as *sparse* (SAEs)*, denoising* (DAEs) and *variational autoencoders* (VAEs) [41–43]. Among them, SAEs, which strive to extract discriminative features avoiding overfitting, are widely used to identify the most relevant and comprehensive set of features for the specific application [44]. Moreover, multiple pretrained AEs can be stacked to form a multiple-hidden-layer ANN, called Stacked Sparse Autoencoder (SSAE), improving the representational and modeling power [45].

In this work, we embed SSAEs for output dimensionality reduction into an IUQ Kriging-based approach, where the Kriging metamodel emulates each of the SSAE extracted features. Up to the authors knowledge, SSAEs (and, in general, non-linear dimensionality reduction techniques) have not yet been applied with Kriging metamodeling in a Bayesian IUQ problem. The rationale for using AEs is related to their capability to: (i) work with raw data without pre-processing; and (ii) deal with nonlinearities [37]. The effectiveness of the proposed approach is shown by its application within a Bayesian IUQ framework aimed at determining the input parameters' PDFs of a RELAP5-3D model of the PERSEO (In-Pool Energy Removal System for Emergency Operation) facility located at SIET laboratory (Piacenza, Italy), for which time series measurements are available. An SSAE is applied to reduce the dimensionality of the problem and fast-running Kriging metamodels are implemented to emulate the RELAP5-3D behavior at a lower computational cost. A comparison between PCA and SSAE for dimensionality reduction is carried out, providing some insights regarding the technical issues associated with the implementation and the use for IUQ.

The remainder of the paper is organized as follows. In Section 2 the formulation of the IUQ problem is presented. Section 3 illustrates the proposed approach. The case study regarding the PERSEO experimental facility and the RELAP5-3D model are introduced in Section 4. Section 5 displays the IUQ results for the proposed approach applied to the case study of Section 4 and provides a comparison between PCA and SSAE. Section 6 concludes the work.

2 The IUQ problem formulation

Let $\boldsymbol{y}^{\boldsymbol{E}}(\boldsymbol{x})$ be a measured experimental quantity and $\boldsymbol{y}^{\boldsymbol{M}}(\boldsymbol{x},\boldsymbol{\theta})$ the corresponding quantity simulated through a computer model. Let $\boldsymbol{x}=\left[x_1,x_2,\dots,x_q\right]^T$ and $\boldsymbol{\theta}=[\theta_1,\theta_2,\dots,\theta_d]^T$ be the *design variables* and the *calibration parameters*, respectively [14]: design variables are all the observable inputs that describe the conditions or scenarios under which the experiment is carried out (e.g., Initial Conditions ICs and Boundary Conditions BCs) [17,24]; calibration parameters are input to the computer model, but they are unknown or not measurable in the physical experiment [17,24]. $\boldsymbol{x}$ are uniquely defined by the experiment and, therefore, known a priori [24]. The objective of the IUQ is to determine the PDF associated with $\boldsymbol{\theta}$.

The relationship between $\boldsymbol{y}^{\boldsymbol{E}}(\boldsymbol{x})$ and $\boldsymbol{y}^{\boldsymbol{M}}(\boldsymbol{x},\boldsymbol{\theta})$ can be described by the *model updating equation* [24]:

$$\boldsymbol{y}^{\boldsymbol{E}}(\boldsymbol{x})=\boldsymbol{y}^{\boldsymbol{M}}(\boldsymbol{x},\boldsymbol{\theta})+\boldsymbol{\delta}(\boldsymbol{x})+\boldsymbol{\epsilon} \tag{1}$$

where $\boldsymbol{\delta}(\boldsymbol{x})$ and $\boldsymbol{\epsilon}$ are the *model discrepancy* (also called *model inadequacy*) and the *measurement error* (also called *observation error*), respectively. $\boldsymbol{\delta}(\boldsymbol{x})$ is due to correlations' inaccuracy or numerical approximations in $\boldsymbol{y^M}(\boldsymbol{x},\boldsymbol{\theta})$, whereas $\boldsymbol{\epsilon}\sim\boldsymbol{N}(\boldsymbol{\mu},\boldsymbol{I}\sigma^2_{exp})$ is an additive measurement error that is commonly assumed to be Gaussian-distributed. The specification of $\boldsymbol{\delta}(\boldsymbol{x})$ requires complex considerations on the model and is ignored in the current work; thus, the model updating equation reduces to:

$$\boldsymbol{y^E}(\boldsymbol{x}) = \boldsymbol{y^M}(\boldsymbol{x},\boldsymbol{\theta}) + \boldsymbol{\epsilon} \tag{2}$$

For a comprehensive discussion about model discrepancy, refer to [14,22,24].

2.1 The Bayesian formulation of the inverse UQ problem

The purpose of IUQ is to quantify the posterior PDF of the calibration parameters $\boldsymbol{\theta}$, $p(\boldsymbol{\theta}|\boldsymbol{y^E})$ i.e., the distribution after the experimental data is observed. According to Bayes rule, this can be calculated as:

$$p(\boldsymbol{\theta}|\boldsymbol{y^E}) = \frac{p(\boldsymbol{y^E}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\boldsymbol{y^E}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}} \tag{3}$$

where $p(\boldsymbol{\theta})$ is the prior PDF and the likelihood function $p(\boldsymbol{y^E}|\boldsymbol{\theta})$ is the probability of observing the experimental data $\boldsymbol{y^E}$, given a particular value of $\boldsymbol{\theta}$.

Let $\boldsymbol{y^E}(\boldsymbol{x}) = [y^E_1(\boldsymbol{x}),\dots,y^E_p(\boldsymbol{x})]$ and $\boldsymbol{y^M}(\boldsymbol{x},\boldsymbol{\theta}) = [y^M_1(\boldsymbol{x},\boldsymbol{\theta}),\dots,y^M_p(\boldsymbol{x},\boldsymbol{\theta})]$ be p-dimensional vectors; if the measurement error is assumed zero-mean Gaussian-distributed (i.e., $\boldsymbol{\epsilon}\sim\boldsymbol{N}(\boldsymbol{0},\boldsymbol{I}\sigma^2_{exp})$), the likelihood function can be derived from equation (2):

$$p(\boldsymbol{y}^E|\boldsymbol{\theta}) = \prod_{i=1}^{N_{exp}} \frac{1}{\left(\sqrt{2\pi}\right)^p\sqrt{\left|\boldsymbol{I}\sigma^2_{exp}\right|}} \exp\left[-\frac{1}{2}[\boldsymbol{y^E}(\boldsymbol{x_i}) - \boldsymbol{y^M}(\boldsymbol{x_i},\boldsymbol{\theta})]^T \left(\boldsymbol{I}\sigma^2_{exp}\right)^{-1} [\boldsymbol{y^E}(\boldsymbol{x_i}) - \boldsymbol{y^M}(\boldsymbol{x_i},\boldsymbol{\theta})]\right] \tag{4}$$

where N_{exp} is the number of independent experimental measurements and $\boldsymbol{I}\sigma^2_{exp}$ is the $p\times p$ covariance matrix of the measurement error. The denominator in the right-hand side of expression (3) is usually analytically intractable; a Markov Chain Monte Carlo (MCMC) algorithm can be adopted to solve this problem and the MCMC samples are, then, used to reconstruct the posterior PDF. MCMC algorithms typically require a considerable number (sometimes hundreds of thousands) of code runs to evaluate $\boldsymbol{y^M}(\boldsymbol{x},\boldsymbol{\theta})$, which can be computationally intensive. To address this issue, Kriging metamodeling is a very popular choice, since it also provides an estimation of the metamodel uncertainty. Assuming that a Kriging metamodel is used to emulate $\boldsymbol{y^M}(\boldsymbol{x},\boldsymbol{\theta})$, the posterior PDF becomes:

$$p(\boldsymbol{\theta}|\boldsymbol{y^E}) \propto p(\boldsymbol{\theta})\cdot\prod_{i=1}^{N_{exp}} \frac{1}{\left(\sqrt{2\pi}\right)^p\sqrt{|\boldsymbol{\Sigma}|}} \exp\left[-\frac{1}{2}[\boldsymbol{y^E}(\boldsymbol{x_i}) - \hat{\boldsymbol{y}}(\boldsymbol{x_i},\boldsymbol{\theta})]^T \boldsymbol{\Sigma}^{-1} [\boldsymbol{y^E}(\boldsymbol{x_i}) - \hat{\boldsymbol{y}}(\boldsymbol{x_i},\boldsymbol{\theta})]\right] \tag{5}$$

where $\widehat{\boldsymbol{y}}(\boldsymbol{x},\boldsymbol{\theta})$ is the Kriging prediction, $\boldsymbol{\Sigma} = \boldsymbol{I}\sigma_{exp}^2 + \boldsymbol{\Sigma}_{MM}$ is the likelihood covariance matrix given by the sum of the covariance matrix of the measurement error $\boldsymbol{I}\sigma_{exp}^2$ and the covariance matrix of the metamodel uncertainty $\boldsymbol{\Sigma}_{MM}$.

3 Proposed IUQ Approach

To perform IUQ within a Bayesian framework, we propose an approach that comprises two main steps:

1. Dimensionality reduction and Kriging metamodeling (Section 3.1);
2. Bayesian inference (by MCMC sampling) (Section 3.2).

3.1 Dimensionality reduction and Kriging Metamodeling

In the context of Bayesian IUQ, the objective of the present Section is to illustrate how to build a metamodel for emulating the time-dependent scalar quantity $y^M(\boldsymbol{x},\boldsymbol{\theta};t)$ computed by a BE code. Without loss of generality, let us assume that only a single experimental time series measurement is available for the Bayesian IUQ (i.e., $N_{exp} = 1$, as it is in the case study illustrated in Section 4); then, the dependence of the forward model $y^M(\boldsymbol{x},\boldsymbol{\theta};t)$ on $\boldsymbol{x}$ is absorbed into the definition of $y^M = y^M(\boldsymbol{\theta};t)$. If we assume that $y^M(\boldsymbol{\theta};t)$ is ticked at p different pre-defined time instances (i.e., $[y^M(\boldsymbol{\theta};t_1),\dots,y^M(\boldsymbol{\theta};t_p)]$), the time-dependent scalar output can be treated as multivariate (vectorial), i.e., $\boldsymbol{y}^{\boldsymbol{M}}(\boldsymbol{\theta}) = [y^M(\boldsymbol{\theta};t_1),\dots,y^M(\boldsymbol{\theta};t_p)]$. Let $\boldsymbol{\Theta} = [\boldsymbol{\theta}^{(1)},\dots,\boldsymbol{\theta}^{(m)}]$ be the Design Of Experiments (DOEs) and $\boldsymbol{Y} = \left[\boldsymbol{y}^{(1)},\dots,\boldsymbol{y}^{(m)}\right]$ the $p \times m$ matrix that contains the corresponding m p-dimensional BE model responses. A way to handle this data is to transform the p-dimensional output $\boldsymbol{y} \in \mathbb{R}^p$ into a reduced p^*-dimensional features space $Z \subset \mathbb{R}^{p^*}$ (with $p^* \ll p$) through a dimensionality reduction technique (e.g., PCA or SSAEs) and, then, build p^* separate independent metamodels that emulate the p^*extracted features. Actually, it is also possible to build a p^*-dimensional multi-output surrogate model, either by Kriging or ANN, especially convenient with PCA since the transformed variables are uncorrelated.

In this work, the m BE model responses contained in $\boldsymbol{Y}$ are mapped onto $Z \subset \mathbb{R}^{p^*}$ and arranged in the $p^* \times m$ features matrix $\boldsymbol{Z} = [\boldsymbol{z}^{(1)},\dots,\boldsymbol{z}^{(m)}]$. Then, each independent metamodel is trained to emulate each feature using the m input-output training patterns (i.e., $\boldsymbol{\Theta} = [\boldsymbol{\theta}^{(1)},\dots,\boldsymbol{\theta}^{(m)}]$ and the corresponding m transformed model responses $\left[z_j^{(1)},\dots,z_j^{(m)}\right]$, with $j = 1,2,\dots,p^*$). Figure 1 shows a schematic diagram of the metamodel approach adopted.

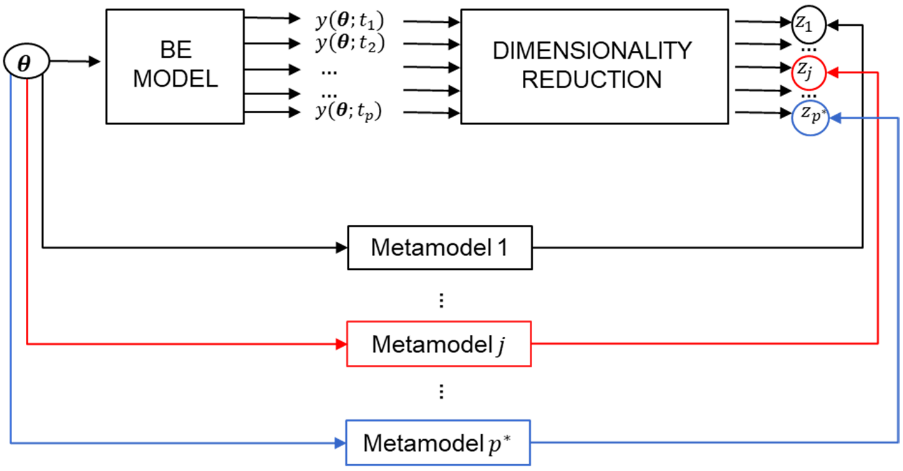


Figure 1. A schematic diagram of the metamodel approach adopted for dimensionality reduction.

Unlike other metamodeling techniques, Kriging provides the uncertainty associated with each prediction, which makes such class of metamodels particularly suitable for Bayesian IUQ since this uncertainty directly enters the likelihood formulation (see Eq. (16) in Section 3.2). Among the different methods proposed to perform dimensionality reduction, we present a method based on SSAEs that is eventually compared to PCA, providing some insights regarding the technical issues associated with the implementation and the use within the proposed Bayesian IUQ approach.

3.1.1 Sparse Autoencoders

An *autoencoder* (AE) is a type of Artificial Neural Network (ANN) that is designed to learn a new low-dimensional latent representation of the data by trying to reconstruct the input data [36]. It is composed of an *encoder* network, that transforms high dimensional data into a lower-dimensional representation called *features*, and a *decoder* network that recovers the original data from the features [37] (Figure 2).

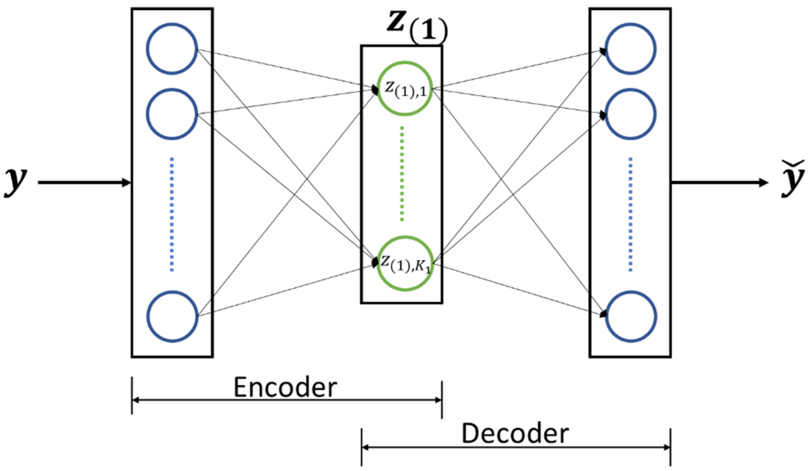


Figure 2. Structure of a basic SAE

The encoder transforms a p-dimensional vector $\boldsymbol{y}$ into its K_1-dimensional hidden representation $\boldsymbol{z}_{(1)} = [z_{(1),1}, z_{(1),2}, \dots z_{(1),K_1}]$:

$$\boldsymbol{z}_{(1)} = f(\boldsymbol{W}_{(1)}\boldsymbol{y} + \boldsymbol{b}_{(1)}) \tag{6}$$

where $z_{(l),j}$ is the j^{th} neuron output of the l^{th} hidden layer (i.e., 1 in the case of basic SAE), f, $\boldsymbol{W}_{(1)}$, $\boldsymbol{b}_{(1)}$ are the encoder activation function, the weight matrix and the bias vector, respectively. The decoder maps the hidden representation back into $\breve{\boldsymbol{y}}$ (i.e., the reconstruction of $\boldsymbol{y}$):

$$\breve{\boldsymbol{y}} = g(\boldsymbol{W}_{(2)}\boldsymbol{z}_{(1)} + \boldsymbol{b}_{(2)}) \tag{7}$$

where g, $\boldsymbol{W}_2$, $\boldsymbol{b}_2$ are the decoder activation function, the weight matrix and the bias vector, respectively.

The SAE, which is a variant of the AE, imposes sparsity constraints on the hidden neurons to encourage the extraction of discriminative features [36,44]. The SAE training (i.e., the determination of $\boldsymbol{W}_{(1)}$, $\boldsymbol{b}_{(1)}$, $\boldsymbol{W}_{(2)}$, $\boldsymbol{b}_2$) is carried out through the minimization of the following cost function:

$$E = R_{error} + \beta R_{sparse} + \lambda R_{L2} \tag{8}$$

where R_{error}, R_{sparse} and R_{L2} are the reconstruction error, the sparsity regularization and the L_2 regularization terms, respectively, whereas β and λ quantify the relative importance of the terms in the cost function. The reconstruction error R_{error} allows quantifying the accuracy of the SAE in the reconstruction of the input vectors:

$$R_{error} = \frac{1}{m_{train}} \sum_{i=1}^{m_{train}} \left\| \boldsymbol{y}^{(i)} - \breve{\boldsymbol{y}}^{(i)} \right\|^2 \tag{9}$$

where m_{train} is the number of training patterns.

A hidden layer's neuron is considered "active" when its value is large (i.e., close to 1.0, in case of sigmoid activation function), and "inactive" when its value is small (i.e., close to 0.0, in case of sigmoid activation function). Let $\hat{\rho}_j$ be the average activation of the j^{th} hidden neuron on the training dataset (i.e., for $\boldsymbol{y}^{(i)} \in \boldsymbol{Y}, i = 1,2, \dots, m_{train}$):

$$\hat{\rho}_j = \frac{1}{m_{train}} \sum_{i=1}^{m_{train}} z_{(1),j}^{(i)} \tag{10}$$

where $z_{(1),j}^{(i)}$ is the j^{th} neuron output of the i^{th} hidden representation $\boldsymbol{z}_{(1)}^{(i)} = \left[z_{(1),1}^{(i)}, \dots, z_{(1),j}^{(i)}, \dots, z_{(1),K_1}^{(i)}\right]$, $j = 1, \dots, K_1,\ i = 1,2, \dots, m_{train}$. It has been shown that the extraction of discriminative features $\boldsymbol{z}_{(\boldsymbol{1})}$ is favored by requiring the sparsity of the AE [44], which imposes the neurons to be inactive most of the time (i.e., all the SAEs hidden neurons are characterized by a small value of $\hat{\rho}_j$, e.g., $\hat{\rho}_j = \rho = 0.05$). To this aim, the sparsity regularization term, R_{sparse}, is added to the cost function to impose such a sparsity constraint. R_{sparse} penalizes $\hat{\rho}_j$ deviating from ρ by using the Kullback-Leibler (KL) divergence function:

$$R_{sparse} = \sum_{j=1}^{K_1} KL(\rho \parallel \hat{\rho}_j) = \sum_{j=1}^{K_1} \left[\rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j}\right] \tag{11}$$

It is worth noting that $KL(\rho \parallel \hat{\rho}_j) = 0$ if $\hat{\rho}_j = \rho$, and it increases monotonically as $\hat{\rho}_j$ diverges from ρ. During the training phase, the output value of the hidden neurons may be lowered by increasing the weight values $\boldsymbol{W}_{(1)}$ making R_{sparse} to be small [41]. To prevent it from happening, the R_{L2} term is added to the cost function:

$$R_{L2} = \frac{1}{2} \|\boldsymbol{W}\| \tag{12}$$

where $\boldsymbol{W}$ is the SAE weight matrix.

3.1.2 Stacked Sparse Autoencoders

Training non-linear autoencoders with multiple hidden layers is a difficult task [37]. To tackle this problem, Hinton and Salakhutdinov [37] proposed the breakthrough approach adopted in this work, which consists of a *pre-training phase* and a *fine-tuning phase*. Let us consider an L-hidden-layer SAE (Figure 3):

1. The pre-training phase regards the consecutive training of L basic SAEs. Initially, the first basic SAE is trained using the input vectors $\boldsymbol{y}^{(i)} \in \boldsymbol{Y}$; then, the corresponding extracted features $\boldsymbol{z}_{(1)}^{(i)}$ are used as training input vectors for the next basic SAE, which transforms $\boldsymbol{z}_{(1)}^{(i)}$ into $\boldsymbol{z}_{(2)}^{(i)}$ (Figure 3a). This procedure is repeated until the last SAE is trained.
2. Then, the SSAE is built by stacking all the basic SAE [46] (Figure 3b).

3. In the fine-tuning phase, the SSAE obtained from the pre-training phase is fine-tuned using the backpropagation of error derivatives [47].

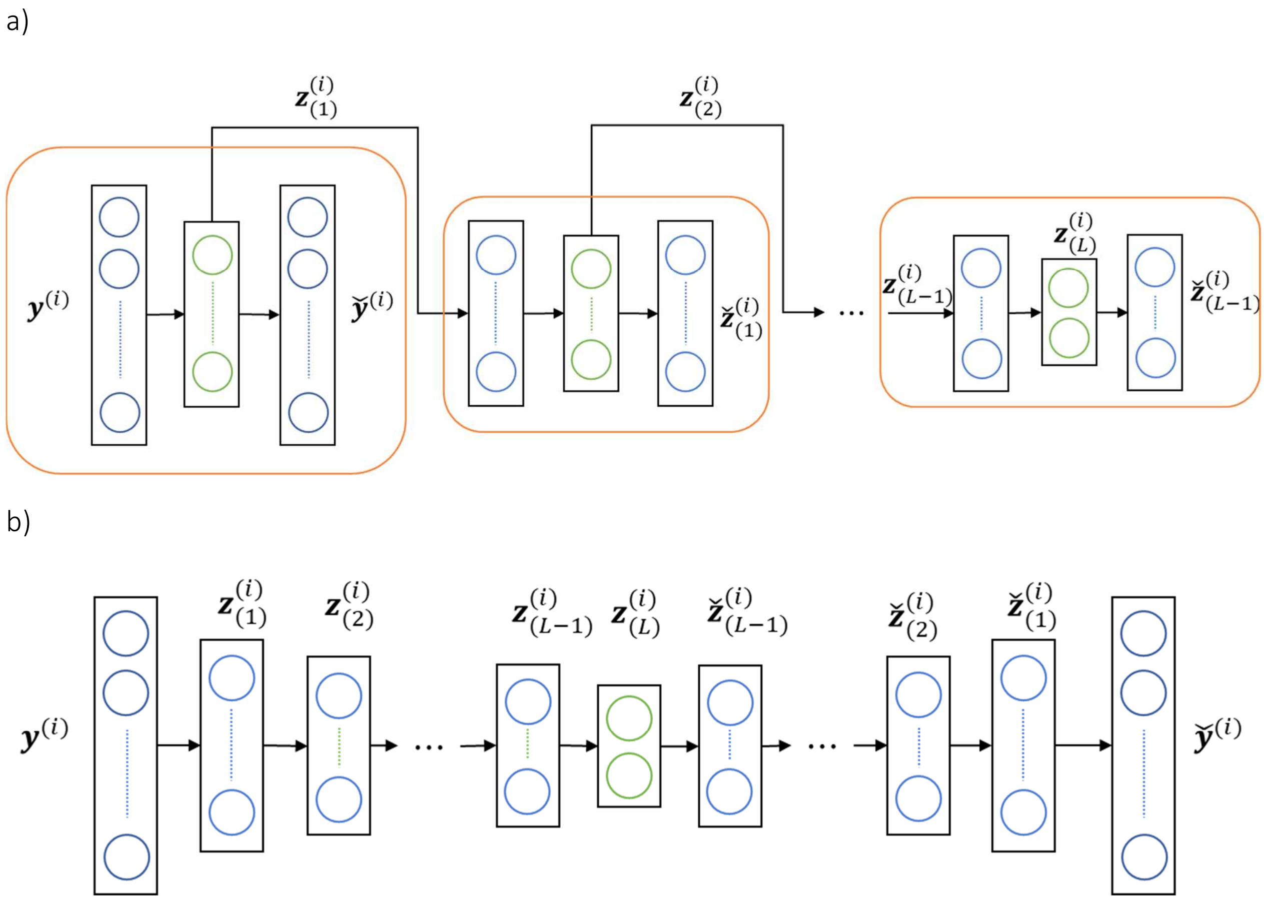

a)
$\boldsymbol{z}_{(1)}^{(i)}$
$\boldsymbol{z}_{(2)}^{(i)}$
$\boldsymbol{y}^{(i)}$
$\check{\boldsymbol{y}}^{(i)}$
$\check{\boldsymbol{z}}_{(1)}^{(i)}$
…
$\boldsymbol{z}_{(L-1)}^{(i)}$
$\boldsymbol{z}_{(L)}^{(i)}$
$\check{\boldsymbol{z}}_{(L-1)}^{(i)}$
b)
$\boldsymbol{y}^{(i)}$
$\boldsymbol{z}_{(1)}^{(i)}$
$\boldsymbol{z}_{(2)}^{(i)}$
…
$\boldsymbol{z}_{(L-1)}^{(i)}$
$\boldsymbol{z}_{(L)}^{(i)}$
$\check{\boldsymbol{z}}_{(L-1)}^{(i)}$
…
$\check{\boldsymbol{z}}_{(2)}^{(i)}$
$\check{\boldsymbol{z}}_{(1)}^{(i)}$
$\check{\boldsymbol{y}}^{(i)}$

Figure 3. Steps of the pre-training phase for an L-hidden-layer SAE: (a) sequential training of L basic SAEs; (b) stacking of the basic SAEs.

Once the SSAE training is completed, its encoder is used to perform dimensionality reduction. The encoder transforms a generic time series $\boldsymbol{y}^{(i)}$ into $\boldsymbol{z}_{(\boldsymbol{L})}^{(\boldsymbol{i})}$, that is a K_L-dimensional vector, where K_L is the number of neurons of the innermost layer.

3.1.3 SSAE performance assessment

Given a set of DOE points $\boldsymbol{\Theta} = [\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(m)}]$, typically built by a Latin Hypercube Sampling (LHS) [48] according to the ranges of the prior distributions, $\boldsymbol{\Theta}$ are simulated through the BE model and the results collected into the $p \times m$ data matrix $\boldsymbol{Y} = \left[\boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(m)}\right]$. The m simulated time series collected in $\boldsymbol{Y}$ are split into two groups: $\boldsymbol{Y}^{TRAIN}$ and $\boldsymbol{Y}^{TEST}$. The training set contains m_{train} time series stored in the $p \times m_{train}$ data matrix $\boldsymbol{Y}^{TRAIN}$, whereas the test set contains $(m - m_{train})$ time series stored in the $p \times (m - m_{train})$ data matrix $\boldsymbol{Y}^{TEST}$. The training of the SSAE starts by defining its architecture (i.e., the number of hidden layers L and the number of neurons per each layer, that is, $K_1, \dots K_L$) and setting the hyperparameters values (i.e., $\rho,\ \beta,\ \lambda$); then, L basic SAEs are trained, stacked and fine-tuned according to the procedure described in Section 3.1.2, and the fine-tuned SSAE is obtained. The SSAE capability in reconstructing time series that differ from the training one is tested on $\boldsymbol{Y}^{TEST}$ through the reconstruction error:

$$R_{error} = \frac{1}{m - m_{train}} \sum_{i=1}^{m-m_{train}} \left\| \boldsymbol{y}_{test}^{(i)} - \breve{\boldsymbol{y}}_{test}{}^{(i)} \right\|^2 \tag{13}$$

Once the SSAE training is completed, K_L independent Kriging metamodels are built (one for each component of $\boldsymbol{z}_{(\boldsymbol{L})}$) using the m inputs $\boldsymbol{\Theta} = \left[\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(m)}\right]$ and the respective transformed time series $\boldsymbol{Z}_{(\boldsymbol{L})} = \left[\boldsymbol{z}_{(\boldsymbol{L})}^{(1)}, \dots, \boldsymbol{z}_{(\boldsymbol{L})}^{(m)}\right]$.

On the other hand, in order to assess the predictive accuracy of each of the $p^* = K_L$ Kriging metamodels, the normalized Leave One Out Cross Validation (LOOCV) error is computed as follows (refer to Appendix C for further details):

$$\epsilon_{LOOCV,j} = \frac{\frac{1}{m}\sum_{i=1}^{m}\left(z_j^{(i)} - \hat{z}_{(-i)}^{MM}\left(\boldsymbol{\theta}^{(i)}\right)\right)^2}{\frac{1}{m}\sum_{i=1}^{m}\left(z_j^{(i)} - \overline{z}_j\right)^2} \tag{14}$$

where m is the Kriging training dataset size, $j = 1 \dots, p^*$, $\hat{z}_{(-i)}^{MM}\left(\boldsymbol{\theta}^{(i)}\right)$ is the Kriging metamodel obtained using all the points of $\boldsymbol{\Theta}$, except $\boldsymbol{\theta}^{(i)}$, $\overline{z}_j = \frac{1}{m}\sum_{i=1}^{m} z_j^{(i)}$. It should be noticed that in (14), the LOOCV error (i.e., $\frac{1}{m}\sum_{i=1}^{m}\left(z_j^{(i)} - \hat{z}_{(-i)}^{MM}\left(\boldsymbol{\theta}^{(i)}\right)\right)^2$) is normalized with respect to the training output sample variance (i.e., $\frac{1}{m}\sum_{i=1}^{m}\left(z_j^{(i)} - \overline{z}_j\right)^2$).

The setting of the SSAE hyperparameters (i.e., ρ, β, λ) is performed by trial-and-error, considering the values of ϵ_{LOOCV} for each metamodel and R_{error}. When a SSAE is trained, the corresponding ϵ_{LOOCV} errors are evaluated and, if acceptable, the SSAE is retained; otherwise, different hyperparameters are set and the entire procedure is repeated. There is no predefined threshold below which R_{error} and ϵ_{LOOCV} are considered acceptable; further details about this last point are given in Section 5.

3.2 Bayesian inference (by MCMC sampling)

According to Ref. [33], performing the IUQ in the original space (i.e., in $\Upsilon \subset \mathbb{R}^p$) through the reconstruction of each Kiging prediction $\hat{\boldsymbol{z}}_{(\boldsymbol{L})}^{\boldsymbol{MM}}(\boldsymbol{\theta})$ may cause convergence issues in the MCMC sampling, if any combination of input parameters $\boldsymbol{\theta}$ does not make the reconstructed time series $\breve{\boldsymbol{y}}(\boldsymbol{\theta})$ agree with the experimental data $\boldsymbol{y}^{\boldsymbol{E}}$. For this reason, in the present research $\boldsymbol{y}^{\boldsymbol{E}}$ is transformed into the feature subspace (obtaining $\boldsymbol{z}^{E}$) and the IUQ is carried out in such reduced space. The prior distribution $p(\boldsymbol{\theta})$ is typically set on the basis of expert judgement and, in the case of scarce knowledge about $\boldsymbol{\theta}$, the prior ranges are taken as large as possible, letting the data speak for itself. The posterior $p(\boldsymbol{\theta}|\boldsymbol{z}^{\boldsymbol{E}})$ is proportional to $p(\boldsymbol{\theta})$ multiplied by the likelihood $p(\boldsymbol{z}^{\boldsymbol{E}}|\boldsymbol{\theta})$:

$$p(\boldsymbol{\theta}|\boldsymbol{z}^{\boldsymbol{E}}) \propto p(\boldsymbol{\theta})p(\boldsymbol{z}^{\boldsymbol{E}}|\boldsymbol{\theta}) \tag{15}$$

The challenging objective addressed in this Section is to formulate the likelihood $p(\boldsymbol{z}^{\boldsymbol{E}}|\boldsymbol{\theta})$ in the case of non-linear dimensionality reduction of the output (e.g., by SSAE). In this work, we derive an expression for the likelihood $p(\boldsymbol{z}^{\boldsymbol{E}}|\boldsymbol{\theta})$ by propagating $\boldsymbol{y}^{\boldsymbol{E}} \sim \boldsymbol{N}(\boldsymbol{y}(\boldsymbol{\theta}), \boldsymbol{I}\sigma_{exp}^2)$ through the SSAE encoder by means of an Extended Kalman Filter (EKF) [49]. The expression of $p(\boldsymbol{z}^{\boldsymbol{E}}|\boldsymbol{\theta})$, in the case of a single independent experimental measurement (i.e., $N_{exp} = 1$), results:

$$p(\boldsymbol{z}^{E}|\boldsymbol{\theta}) = \frac{1}{(\sqrt{2\pi})^{p^*}\sqrt{|\boldsymbol{\Sigma}_{\boldsymbol{exp}}(\boldsymbol{\theta}) + \boldsymbol{\Sigma}_{Kriging}(\boldsymbol{\theta})\,|}} exp\left[-\frac{1}{2}\left[\boldsymbol{z}^{E} - \hat{\boldsymbol{z}}_{(\mathrm{L})}^{MM}(\boldsymbol{\theta})\right]^{T} (\boldsymbol{\Sigma}_{\boldsymbol{exp}}(\boldsymbol{\theta}) + \boldsymbol{\Sigma}_{Kriging}(\boldsymbol{\theta}))^{-1} \left[\boldsymbol{z}^{E} - \hat{\boldsymbol{z}}_{(\mathrm{L})}^{MM}(\boldsymbol{\theta})\right]\right] \tag{16}$$

where $\hat{\boldsymbol{z}}_{(\boldsymbol{L})}^{\boldsymbol{MM}}(\boldsymbol{\theta})$ is the Kriging prediction in the reduced space; $\boldsymbol{\Sigma}_{\boldsymbol{exp}}(\boldsymbol{\theta})$ is the experimental uncertainty covariance matrix in the p^*-dimensional reduced space; and $\boldsymbol{\Sigma}_{\boldsymbol{Kriging}}(\boldsymbol{\theta})$ is the covariance matrix associated with the Kriging prediction uncertainty, i.e., a $p^* \times p^*$ matrix having the mean squared errors of each feature prediction as diagonal entries:

$$\boldsymbol{\Sigma}_{\boldsymbol{Kriging}} = \begin{bmatrix} \sigma_{z_1(\theta)}^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_{z_{p^*}(\theta)}^2 \end{bmatrix} \tag{17}$$

Details on the hypothesis adopted for the derivation of $p(\boldsymbol{z}^{\boldsymbol{E}}|\boldsymbol{\theta})$ are given in Appendix C. Once the likelihood is formulated, a MCMC algorithm can be adopted to sample from $p(\boldsymbol{\theta}|\boldsymbol{z}^{\boldsymbol{E}})$ and the posterior samples can be used to reconstruct the posterior PDF.

4 Case Study

The proposed IUQ approach is applied to a TH input model for the RELAP5-3D code [50] developed by Politecnico di Torino [51] for the PERSEO test facility, a full-scale facility with a maximum power of around

20 MW, designed to assess the performance and the efficiency of a new in-pool heat exchanger for decay heat removal by natural circulation [52]. The PERSEO test facility, located at SIET laboratories, is sketched in Figure 4. For more details about the facility, please refer to [52,53]. PERSEO Test 7 experimental data have been provided to Politecnico di Torino by ENEA for the participation in the PERSEO benchmark, within the OECD/NEA/CSNI/WGAMA "Status report on thermal-hydraulic passive systems design and safety assessment". Experimental data from PERSEO Test 7 are here used for IUQ.

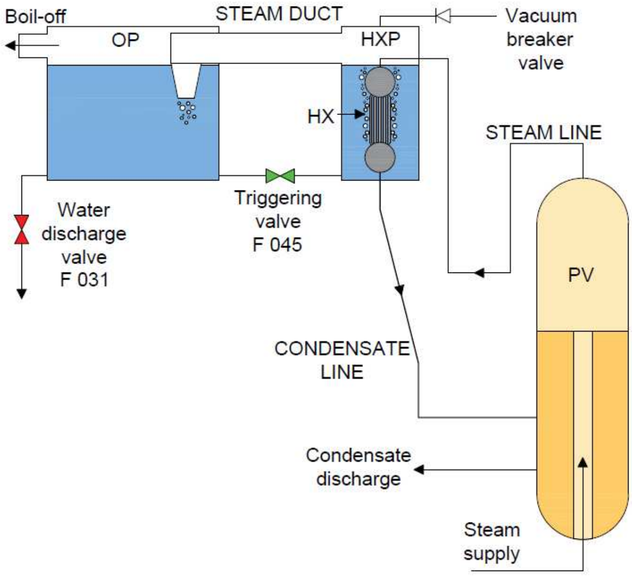


Figure 4. Scheme of the PERSEO facility [51]

The PERSEO experimental campaign consists of nine tests [52,53]. Test 7 comprises two parts: a first part to verify the behavior of the system with two different water levels, and a second part to characterize the long term cooling capability of the system [52]. For our analysis, the measured experimental data of Test 7-Part 2 are used [52]. According to the classification of input parameters made in Section 2, the set of the design variables $\boldsymbol{x}$ are: ICs, BCs and the set of scenario-description inputs (e.g., Triggering Valve TV, opening/closure) [52,54]. Concerning the calibration variables $\boldsymbol{\theta}$, the RELAP5-3D code contains different model input parameters that could be included in the IUQ process. However, in the present analysis, only some of them are considered: in fact, the objective of the present study is not to carry out a complete uncertainty analysis but to show how the IUQ could be performed taking advantage of the adopted methodology. For each parameter, a uniform prior PDF is set: $p(\theta) = \frac{1}{U-L}$ for $\theta \in [L, U]$ and $p(\theta) = 0$ otherwise, where U and L are the upper and lower bounds, respectively (as listed in Table 1). It is worth mentioning that the values reported in Table 1 are rescaled factors (i.e., all the parameters have been normalized with respect to their prior nominal values). For more details about the description of the parameters and the selection of the prior ranges, please refer to [35].

Table 1. The uncertain input parameters selected for the RELAP5-3D model.

θ_i	Parameter (multiplication factor)	Parameter Name	Lower bound	Upper bound
θ_1	Inner fouling factor	Inner_FF	0.5	1.5
θ_2	Outer fouling factor	Outer_FF	1.0	1.5
θ_3	Injector K factor	K_injector	0.5	1.5
θ_4	Sum of the steam line's K factors	K_sum_steam	0.5	1.5
θ_5	Sum of condensate line's K factors	K_sum_condensate	0.5	1.5
θ_6	Diaphragm K factor	K_ diaphragm	0.5	1.5
θ_7	Rockwool thermal conductivity	k_rockwool	1.0	1.5
θ_8	HXP first pipe flow area	A_effective	0.5	1.5

The power exchanged by the HX is used as the experimental data to perform the IUQ, because it conveys a considerable amount of information about the transient (e.g., the heat exchange between the HX tubes and the Heat Exchanger Pool HXP); thus, it is one of the most representative output. Figure 5 shows a comparison between the RELAP5-3D HX exchanged power computed for Test 7-Part 2 using the nominal values of the prior distribution and the corresponding experimental data.

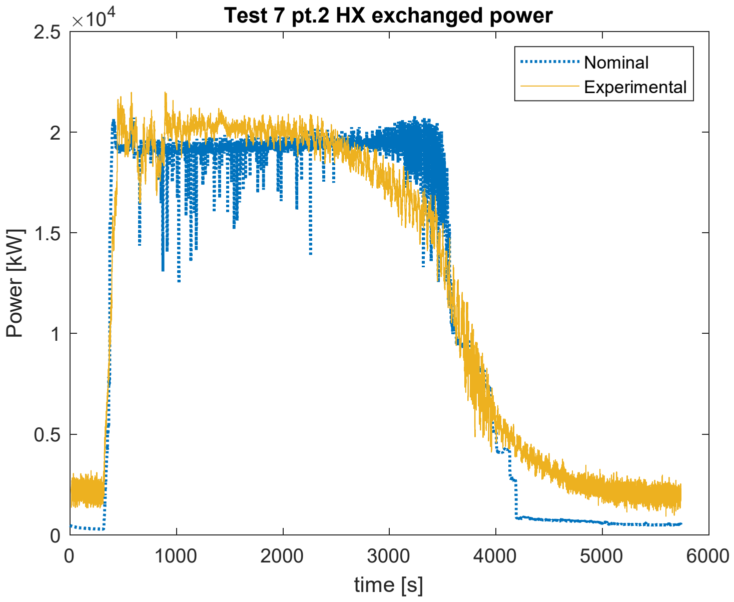


Figure 5. HX exchanged power (Test 7-Part 2). In yellow solid line, the experimental measurement; in blue dotted line, the RELAP5-3D simulation for the nominal values.

Since we consider only a single experiment (i.e., Test 7-Part 2), we absorb the dependence on $\boldsymbol{x}$ of the forward model $\boldsymbol{y}^{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{\theta})$ into the definition of $\boldsymbol{y}^{\boldsymbol{M}} = \boldsymbol{y}^{\boldsymbol{M}}(\boldsymbol{\theta})$ that represents the RELAP5-3D model. Such a model predicts the vector $\boldsymbol{y}^{\boldsymbol{M}} = [y^M(\boldsymbol{\theta}; t_1), \dots, y^M(\boldsymbol{\theta}; t_p)]$, whose entries $y^M(\boldsymbol{\theta}; t)$ are the HX exchanged power during Test 7-Part 2 at times t. A RELAP5-3D simulation, performed using an Intel Core i7-7500U processor, takes around 2 hours. Simulations are carried out at the Energy Department of Politecnico di Torino by faculty members.

4.1 Data description

Let $y^E = \left[y_1^E, \dots, y_p^E\right]^T$ be the available experimental HX exchanged power time series measured during Test 7-Part 2. Latin Hypercube Sampling (LHS) is adopted to build the DOE $\boldsymbol{\Theta} = [\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(m)}]$, that contains $m = 180$ training inputs, according to the prior distributions reported in Table 1. The corresponding RELAP5-3D output realizations $\boldsymbol{y}^{(i)}$ are, then, stored into the $p \times m$ training output data matrix $\boldsymbol{Y} = \left[\boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(m)}\right]$ with $p = 5723$. Figure 6 shows that the ensemble of the $m = 180$ RELAP5-3D output realizations (i.e., time series) contained in $\boldsymbol{Y}$ are affected by oscillations with a small signal-to-noise ratio. We apply the novel IUQ approach, based on SSAEs, on the dataset described here and compare the results with the standard approach based on PCA.

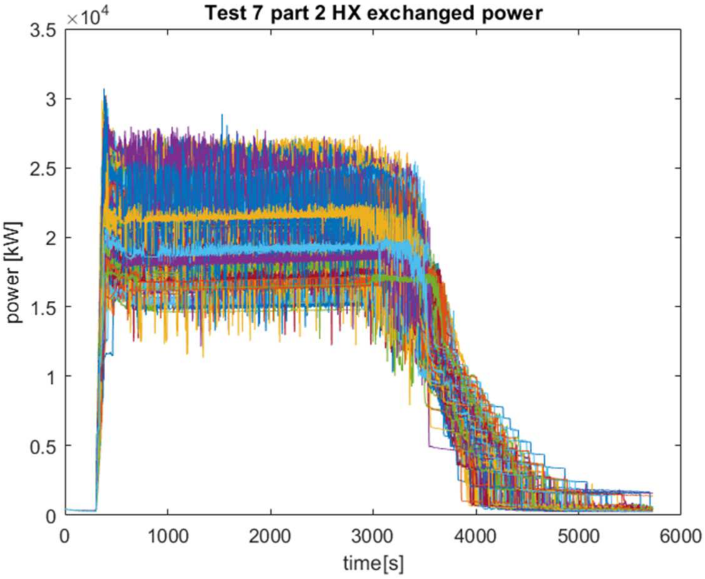


Figure 6. The ensemble of the $\boldsymbol{m} = \mathbf{180}$ RELAP5-3D output realizations contained in $\boldsymbol{Y}$, characterized by a small signal-to-noise ratio.

In this work, the design variables vector $\boldsymbol{x}$ (that defines the conditions under which the PERSEO experiment is carried out) is fixed since $N_{exp} = 1$; for this reason, the trends of the output training data reported in

Figure 6 look very similar. If $N_{exp} > 1$, the trends of the training output could be very different; consequently, the number of features required to map the outputs in the reduced space may increase, whereas the Kriging performances, associated with each feature should be assessed case by case, but this is out of the scope of the present paper.

5 Results

To analyze the effectiveness of the proposed SSAE approach, in this Section we compare the SSAE results with those obtained by a standard PCA-based dimensionality reduction approach of literature [20,31–35]. In particular, Section 5.1 shows 1) how PCA is not capable of dealing with non-filtered (raw) data and 2) proposes some reflections on the main limitations involved by data filtering as a possible solution to deal with such raw data. Moreover, to assess the capability of the SSAE to deal with raw data, in Sections 5.2.1 and 0 the SSAE is fed with non-filtered (raw) and filtered time series, respectively. Section 5.3 shows the results of the forward uncertainty propagation performed through both the metamodel and the RELAP5-3D model.

5.1 PCA-based dimensionality reduction

Let $\boldsymbol{\Theta} = [\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(m)}]$ and $\boldsymbol{Y} = \left[\boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(m)}\right]$ be the $m = 180$ Design of experiment (DOE) points and the corresponding RELAP5-3D model response reported in Section 4.1, respectively. To perform the PCA, the $p \times m$ data matrix $\boldsymbol{Y}$ is centered, obtaining $\boldsymbol{Y}_{centered}$; then, the Singular Value Decomposition (SVD) is carried out for $\boldsymbol{Y}_{centered}$. Further details about PCA are given in Appendix B.1. A number $p^* = 31$ of PCs guarantees a cumulative percentage of variation explained at least to 95%. Then, p^*independent Kriging metamodels are built, one for each feature (i.e., for each PC), using the m input-output training patterns given by $\boldsymbol{\Theta} = [\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(m)}]$ and $\boldsymbol{Z} = \Phi(\boldsymbol{Y} - \overline{\boldsymbol{\mu}}_{\boldsymbol{Y}}) = \begin{bmatrix} z_1^{(1)} & \cdots & z_1^{(m)} \\ \vdots & \ddots & \vdots \\ z_{p^*}^{(1)} & \cdots & z_{p^*}^{(m)} \end{bmatrix}$, where $\overline{\boldsymbol{\mu}}_{\boldsymbol{Y}} = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{y}^{(i)}$ and Φ is the PCA transformation matrix. Each Kriging metamodel is trained to map from $\Theta \subset \mathbb{R}^d$ to $z_j \in \mathbb{R}$ (with $j = 1, \dots, p^*$). To evaluate the capability of PCA of reconstructing the noisy transients, we transform back $\boldsymbol{Z}$ into the original space

$$\breve{\boldsymbol{Y}} = \overline{\boldsymbol{\mu}}_{\boldsymbol{Y}} + \Phi^T \boldsymbol{Z} \tag{18}$$

and compute the reconstruction error R_{error} through Eq. (9). We find that $R_{error} = 2.2647 \times 10^5 kW$.

In the current research, Matérn 5/2 correlation kernel, constant trend function and CV hyperparameter estimation are adopted for each Kriging metamodel, as done in [35] where data filtering is applied on the same training dataset, before PCA. Refer to Appendix A for further details on Kriging metamodeling. In order to assess the predictive accuracy of each of the $p^* = 31$ Kriging metamodels, the normalized Leave One Out Cross Validation (LOOCV) error ϵ_{LOOCV} is computed and reported in Figure 7. It is worth noticing that ϵ_{LOOCV} increases for PCs of higher order, which means that the metamodel prediction capability decreases for higher-order PCs. Likely, the first PCs are those associated with the genuine signal variation (due to physical phenomena); on the contrary, PCs of higher-order relate to the higher noise in comparison with the experimental data of the HX exchanged power predicted by RELAP5-3D model that is significant for the case study proposed. Consequently, training a metamodel for the first PCs is easier since there is an underlying function (given by the physics of the phenomena) that links the inputs of $\boldsymbol{\Theta}$ and the transformed output of $\boldsymbol{Z}$.

According to its definition, an ϵ_{LOOCV} close to 1.0 relates to scarce metamodel predictive capability; indeed, it implies that the LOOCV error of the j^{th} Kriging metamodel

$$\frac{1}{m}\sum_{i=1}^{m}\left(z_j^{(i)} - \hat{z}_{(-i)}^{MM}\left(\boldsymbol{\theta}^{(i)}\right)\right)^2 \tag{19}$$

is of the same order of magnitude of the transformed output sample variance $var[z_j]$:

$$var[z_j] = \frac{1}{m}\sum_{i=1}^{m}\left(z_j^{(i)} - \bar{z}_j\right)^2 \tag{20}$$

Because of the scarce performances that characterize most of the Kriging metamodels (for many of them $\epsilon_{LOOCV} \cong 1.0$ (Figure 7)), they cannot be used to replace the RELAP5-3D model in the IUQ process. This shows that in this context, PCA cannot cope with raw (non-filtered) data; thus, this analysis is not carried out.

Given the limitations of the PCA to cope with raw data (i.e., ϵ_{LOOCV} close to 1.0 for most of the features), a possible solution is to filter the HX power exchanged predicted by RELAP5-3D (noisy) raw time series collected in $\boldsymbol{Y}$ before applying PCA and Kriging metamodels. Although this allows removing the part of output variability due to oscillations, data filtering gives rise to a nontrivial issue, i.e., choosing a proper filtering technique (that should consider the smallest timescale on which physical phenomena take place during the transient). Moreover, although filtering raw data affected by numerical oscillation may appear reasonable, (1) justifying the choice of a particular filtering method rather than another is a nontrivial task, and (2) the IUQ results may be affected by the selection of the specific filtering method adopted. In Ref. [35], we apply a moving median filter (i.e., by the ”movmedian” MATLAB function) on the columns of the same dataset $\boldsymbol{Y}$ before performing PCA and Kriging metamodeling, and finding that a smaller number of PCs (i.e., $p^* = 4$, rather than 31) is able to explain the same percentage of the variance of the dataset (i.e., 95%). These results are reported for completeness in Appendix B.2 and compared, in Section 5.2, to those obtained for the SSAE.

5.2 SSAE-based dimensionality reduction

5.2.1 Non-filtered (raw) data

In this Section, we take advantage of the SSAE properties to perform dimensionality reduction without applying data filtering. The SSAE architecture and the pre-training hyperparameters are set according to Table 2. The hyper-parameters K_1, K_2, K_3 and L are tuned following a trial-and-error approach based on the SSAE performances (i.e., R_{error} and the ϵ_{LOOCV} for each feature) (given that $L = 3$, $p^* = K_3$). More powerful tuning approaches (e.g., extensive grid search and evolutionary optimization) can be used, but at a much higher computational cost.

Table 2. SSAE architecture and hyperparameters (non-filtered data).

Architecture		Hyperparameters (SAE pre-training)	
$\boldsymbol{L}$	**3**	$\boldsymbol{\rho}$	$\mathbf{0.05}$
$\boldsymbol{K_1}$	**200**	$\boldsymbol{\beta}$	**1**
$\boldsymbol{K_2}$	**20**	$\boldsymbol{\lambda}$	$\mathbf{0.001}$
$\boldsymbol{K_3}$	**8**		

Unlike PCA, there is no rule of thumb to define p^*, i.e., K_3, which is here set to $K_3 = 8$. The $m = 180$ simulated time series collected in $\boldsymbol{Y}$ are split into two groups: $\boldsymbol{Y}^{TRAIN}$, that contains $m_{train} = 162$ time series, and $\boldsymbol{Y}^{TEST}$ that contains $m_{test} = m - m_{train} = 18$ time series (i.e., 10% of the available data). $L = 3$ basic SAEs are pre-trained using $\boldsymbol{Y}^{TRAIN}$, with the hyperparameter reported in Table 2; then, they are stacked and fine-tuned according to the procedure described in Section 3.1.2. The fine-tuned SSAE thereby obtained is tested on $\boldsymbol{Y}^{TEST}$ obtaining a reconstruction error equal to $R_{error} = 8.4935 \times 10^5 kW$. A separate independent Kriging is built for each feature and ϵ_{LOOCV} is computed for each of them through (14). Therefore, $K_3 = 8$ independent Kriging metamodels are built adopting the Matérn 5/2 correlation kernel, constant trend function and CV hyperparameter estimation. In this case, R_{error} is computed on $\boldsymbol{Y}^{TEST}$, whereas in the case of PCA R_{error} is computed on $\boldsymbol{Y}$. We can notice that, even though R_{error} is higher, it has the same order of magnitude of that obtained by PCA with filtered and raw data (Table 6). Figure 7 shows that the SSAE, without any user-experience-based data filtering approach, obtains ϵ_{LOOCV} values that are comparable (i.e., between 0.040 and 0.260) to those of PCA applied to filtered data.

Once the SSAE and the Kriging metamodels are trained, the algorithm recalled in Section 3.2 (and presented in Appendix C) is implemented to carry out Bayesian inference. The measurement error standard deviation σ_{exp} is set to $500\ kW$ [53] and an adaptive Metropolis algorithm is applied to generate 8 parallel chains with $1 \cdot 10^5$ iterations. It took almost 16 hours to compute the posterior using an Intel Core i7-7500U processor. According to [13], we post-process the samples by discarding, for each chain, the first half for burn-in to diminish the influence of the starting samples, as a conservative choice. The MCMC convergence is examined through the approach proposed in [13]. The KDE of the posterior marginals PDFs obtained through the SSAE with non-filtered data are displayed in Figure 8. Some summary statistics of the posterior distribution are reported in Table 3, whereas Table 4 shows the correlation among the calibration parameters.

Table 3. Posterior summaries (SSAE with non-filtered data).

$\boldsymbol{\theta_i}$	Parameter	Mean value	Mode	5th percentile	95th percentile
$\boldsymbol{\theta_1}$	Inner_FF	0.91	0.84	0.76	1.06
$\boldsymbol{\theta_2}$	Outer_FF	1.25	1.22	1.04	1.46
$\boldsymbol{\theta_3}$	K_injector	1.25	0.97	0.74	1.48
$\boldsymbol{\theta_4}$	K_sum_steam	1.13	1.49	0.64	1.47
$\boldsymbol{\theta_5}$	K_sum_condensate	0.91	0.61	0.53	1.42
$\boldsymbol{\theta_6}$	K_ diaphragm	1.03	1.35	0.57	1.46
$\boldsymbol{\theta_7}$	k_rockwool	1.27	1.46	1.03	1.48
$\boldsymbol{\theta_8}$	A_effective	0.80	0.84	0.57	0.98

Table 4. Correlation Matrix computed using the MCMC samples (SSAE with non-filtered data).

	$\boldsymbol{\theta_1}$	$\boldsymbol{\theta_2}$	$\boldsymbol{\theta_3}$	$\boldsymbol{\theta_4}$	$\boldsymbol{\theta_5}$	$\boldsymbol{\theta_6}$	$\boldsymbol{\theta_7}$	$\boldsymbol{\theta_8}$
$\boldsymbol{\theta_1}$	1,0000	-	-	-	-	-	-	-
$\boldsymbol{\theta_2}$	0,0758	1,0000	-	-	-	-	-	-
$\boldsymbol{\theta_3}$	-0,0815	-0,4110	1,0000	-	-	-	-	-
$\boldsymbol{\theta_4}$	-0,1261	-0,0720	-0,1443	1,0000	-	-	-	-
$\boldsymbol{\theta_5}$	0,0006	-0,0943	0,0772	-0,1062	1,0000	-	-	-
$\boldsymbol{\theta_6}$	-0,0723	-0,0788	-0,0279	0,0156	0,0249	1,0000	-	-
$\boldsymbol{\theta_7}$	-0,0731	0,0623	-0,1287	-0,0523	-0,0490	-0,0092	1,0000	-
$\boldsymbol{\theta_8}$	0,2947	-0,4725	-0,0484	0,1701	-0,0192	0,0345	-0,0055	1,0000

5.2.2 Filtered data

For comparison purposes, we feed the SSAE with filtered data. The SSAE architecture is set according to the parameters reported in Table 5. Also in this case, a trial-and-error approach based on the SSAE performances (i.e., R_{error} and the ϵ_{LOOCV} for each feature) is adopted.

Table 5. SSAE architecture and hyperparameters (filtered data).

Architecture		Hyperparameters (SAE pre-training)	
$\boldsymbol{L}$	$\mathbf{2}$	$\boldsymbol{\rho}$	$\mathbf{0.05}$
$\boldsymbol{K_1}$	$\mathbf{50}$	$\boldsymbol{\beta}$	$\mathbf{1}$
$\boldsymbol{K_2}$	$\mathbf{5}$	$\boldsymbol{\lambda}$	$\mathbf{0.0001}$

The same train-test split procedure of Section 5.2.1 is implemented on $\boldsymbol{Y}$, and a reconstruction error equal to $R_{error} = 7.9071 \times 10^5 kW$ is obtained. $K_2 = 5$ independent Kriging metamodels are built adopting the Matérn 5/2 correlation kernel, constant trend function and CV hyperparameter estimation. Table 6 and Figure 7 show, respectively, R_{error} and ϵ_{LOOCV} of the Kriging metamodels compared in the cases of: (1) PCA with non-filtered raw data, (2) PCA with filtered data, (3) SSAE with non-filtered raw data and (4) SSAE with filtered data.

Table 6. $\boldsymbol{R_{error}}$ in the case of: PCA with non-filtered data, (2) PCA with filtered data (3) SSAE with non-filtered data and (4) SSAE with filtered data.

Reconstruction error $\boldsymbol{R_{error}}$	
PCA non-filtered data	$\mathbf{2.2647} \times \mathbf{10^5} \boldsymbol{kW}$
PCA filtered data	$\mathbf{2.1308} \times \mathbf{10^5} \boldsymbol{kW}$
SSAE non-filtered data (computed for $\boldsymbol{Y^{TEST}}$)	$\mathbf{8.4935} \times \mathbf{10^5} \boldsymbol{kW}$
SSAE filtered data (computed for $\boldsymbol{Y^{TEST}}$)	$\mathbf{7.9071} \times \mathbf{10^5} \boldsymbol{kW}$

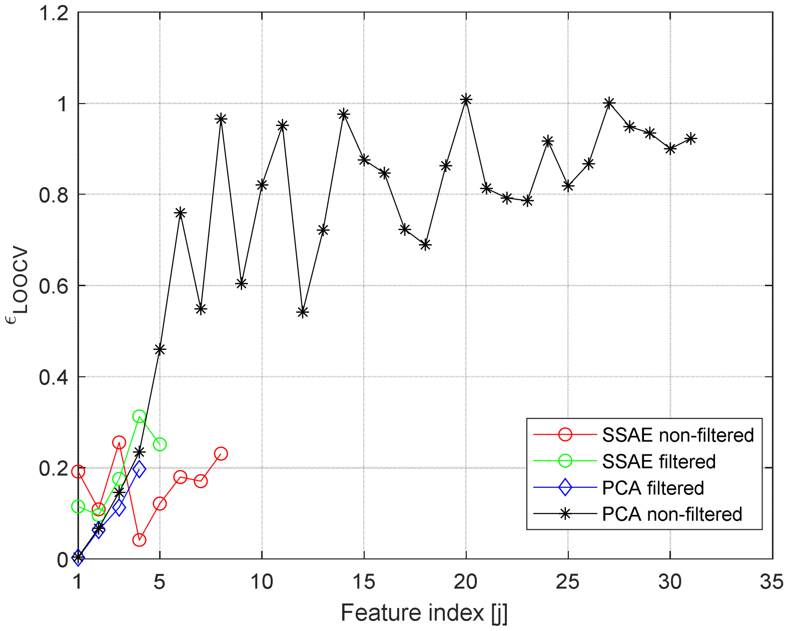


Figure 7. $\boldsymbol{\epsilon_{LOOCV}}$ in the case of PCA with non-filtered data, (2) PCA with filtered data (3) SSAE with non-filtered data and (4) SSAE with filtered data.

Notice that performing data filtering on $\boldsymbol{Y}$ before applying PCA allows reducing: (1) the number of PCs required to have the same cumulative percentage of variation explained (i.e., $p^* = 4$, instead of 31); and (2) the highest value of ϵ_{LOOCV} (that is reduced below 0.20). In contrast, data filtering, for the SSAE architecture and hyperparameter setting proposed in Table 5, does not bring a significant improvement, with respect to the case of SSAE with non-filtered data in terms of R_{error} on test data and ϵ_{LOOCV}. In our case study, output training data have similar trends (as illustrated in Figure 6). If, on the contrary, training time series would have been very different, the number of extracted features (i.e., the number of neurons in the hidden layer) required to get the same SSAE reconstruction error R_{error} could be higher (i.e., the ANN complexity increases). In this case, since we build an independent Kriging metamodel for each extracted feature (as shown Figure 1), the number of Kriging increases and the efficiency of each metamodel should be re-assessed (since it can not be a prior predicted). More sophisticated approaches than the simple trial-and-error (e.g., evolutionary optimization techniques) can be implemented to find the set of SSAE's hyperparameters that minimize both R_{error} and ϵ_{LOOCV}. However, this would be cumbersome considering that: (1) PCA is easier to implement and (2) at lower computational cost, it performs better in terms of R_{error} and ϵ_{LOOCV} with filtered data. Moreover, for PCA the $\boldsymbol{\Sigma}_{exp}$ matrix does not need to be computed at each MCMC iteration (unlike the SSAE case, as shown in Appendix C), and this significantly reduces the computational cost required by the algorithm. In fact, in the case of SSAE with filtered data, the adaptive Metropolis algorithm takes almost $\mathbf{10.5}$ hours for $\mathbf{2 \cdot 10^5}$ iterations on an Intel Core i7-7500U and, considering that the MCMC algorithm developed for the PCA takes $\mathbf{2.5}$ hours for $\mathbf{10^5}$ iterations (on the same processor), the computational cost required by the MCMC algorithm in this case of SSAE is twice that required for PCA. Thus, we can conclude that using the SSAE is conveniently applied to raw data (as expected in real applications), avoiding filtering.

From each chain, the first half of the samples are discarded for burn-in. A common practice adopted to reduce autocorrelation is thinning [13]. It consists in keeping every k^{th} sample from each sequence and discarding the rest: we kept every 2000^{th} (out of $5 \cdot 10^5$), 4000^{th} (out of $1 \cdot 10^5$) and 8000^{th} (out of $2 \cdot 10^5$) sample from each chain for PCA with filtered data, SSAE with raw data and SSAE with filtered data, respectively. The KDE of the posterior marginals PDFs obtained through the SSAE with non-filtered data are displayed in Figure 8. The summary statistics of the posterior distribution are reported in Table 7. The marginal posterior distributions in Figure 8 cannot be used to draw samples, because the calibration parameters are not independent.

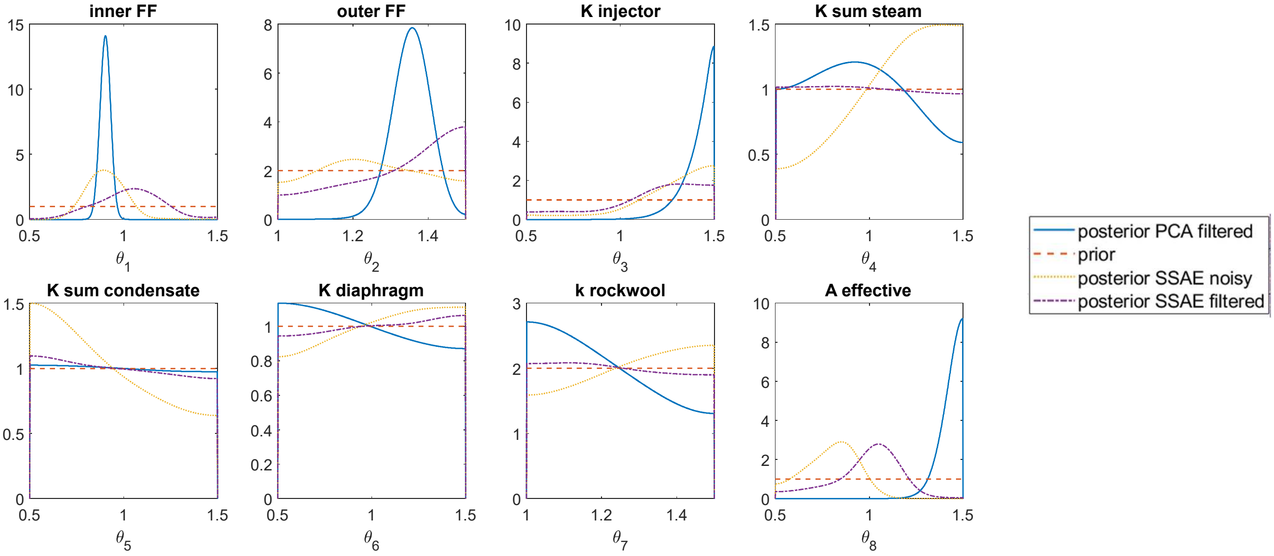


Figure 8. Prior distributions and posteriors' kernel density estimations obtained adopting three different output dimensionality reduction techniques (i.e., PCA with filtered data and SSAE with both filtered and non-filtered data) in the IUQ process.

Table 7. Posterior summaries (SSAE with filtered data).

$\boldsymbol{\theta_i}$	Parameter	Mean value	Mode	5th percentile	95th percentile
$\boldsymbol{\theta_1}$	Inner_FF	1.03	1.18	0.75	1.26
$\boldsymbol{\theta_2}$	Outer_FF	1.32	1.46	1.05	1.49
$\boldsymbol{\theta_3}$	K_injector	1.17	1.24	0.63	1.47
$\boldsymbol{\theta_4}$	K_sum_steam	1.00	1.16	0.55	1.45
$\boldsymbol{\theta_5}$	K_sum_condensate	0.98	0.61	0.54	1.44
$\boldsymbol{\theta_6}$	K_ diaphragm	1.01	1.08	0.55	1.45
$\boldsymbol{\theta_7}$	k_rockwool	1.25	1.00	1.00	1.50
$\boldsymbol{\theta_8}$	A_effective	0.98	1.14	0.63	1.20

Table 8. Correlation Matrix computed using the MCMC samples (SSAE with filtered data).

	$\boldsymbol{\theta_1}$	$\boldsymbol{\theta_2}$	$\boldsymbol{\theta_3}$	$\boldsymbol{\theta_4}$	$\boldsymbol{\theta_5}$	$\boldsymbol{\theta_6}$	$\boldsymbol{\theta_7}$	$\boldsymbol{\theta_8}$
$\boldsymbol{\theta_1}$	1,0000	-	-	-	-	-	-	-
$\boldsymbol{\theta_2}$	0,3468	1,0000	-	-	-	-	-	-
$\boldsymbol{\theta_3}$	-0,1763	-0,1013	1,0000	-	-	-	-	-
$\boldsymbol{\theta_4}$	-0,0319	0,0062	0,0441	1,0000	-	-	-	-
$\boldsymbol{\theta_5}$	-0,0018	-0,0038	-0,0048	-0,0074	1,0000	-	-	-
$\boldsymbol{\theta_6}$	0,0384	0,1051	-0,0768	-0,0130	-0,0158	1,0000	-	-
$\boldsymbol{\theta_7}$	0,0319	-0,0184	0,0331	-0,0270	0,0101	-0,0010	1,0000	-
$\boldsymbol{\theta_8}$	0,4391	0,2914	-0,3216	0,0207	0,0446	0,0317	-0,0491	1,0000

In all the cases examined, the marginal posteriors of $\theta_4, \theta_5, \theta_6, \theta_7$ do not differ very much from their priors and are defined on the same supports of their priors, whereas for θ_1 and θ_8 a significant update can be seen with respect to the priors. This is in line with the results found in [35], where the sensitivity analysis, carried out through first-order Sobol' indices [55], revealed that $\theta_4, \theta_5, \theta_6, \theta_7$ are less influential. In particular, for the PCA-based approach, the marginal posteriors of $\theta_1, \theta_2, \theta_3, \theta_8$ show a significant modification with respect to their priors and the posterior of θ_8 is peaked at the upper bound of the prior, whereas in both SSAE cases the marginal posterior of θ_2 is quite similar to the prior, and the posterior of θ_8 is peaked near the prior mean value.

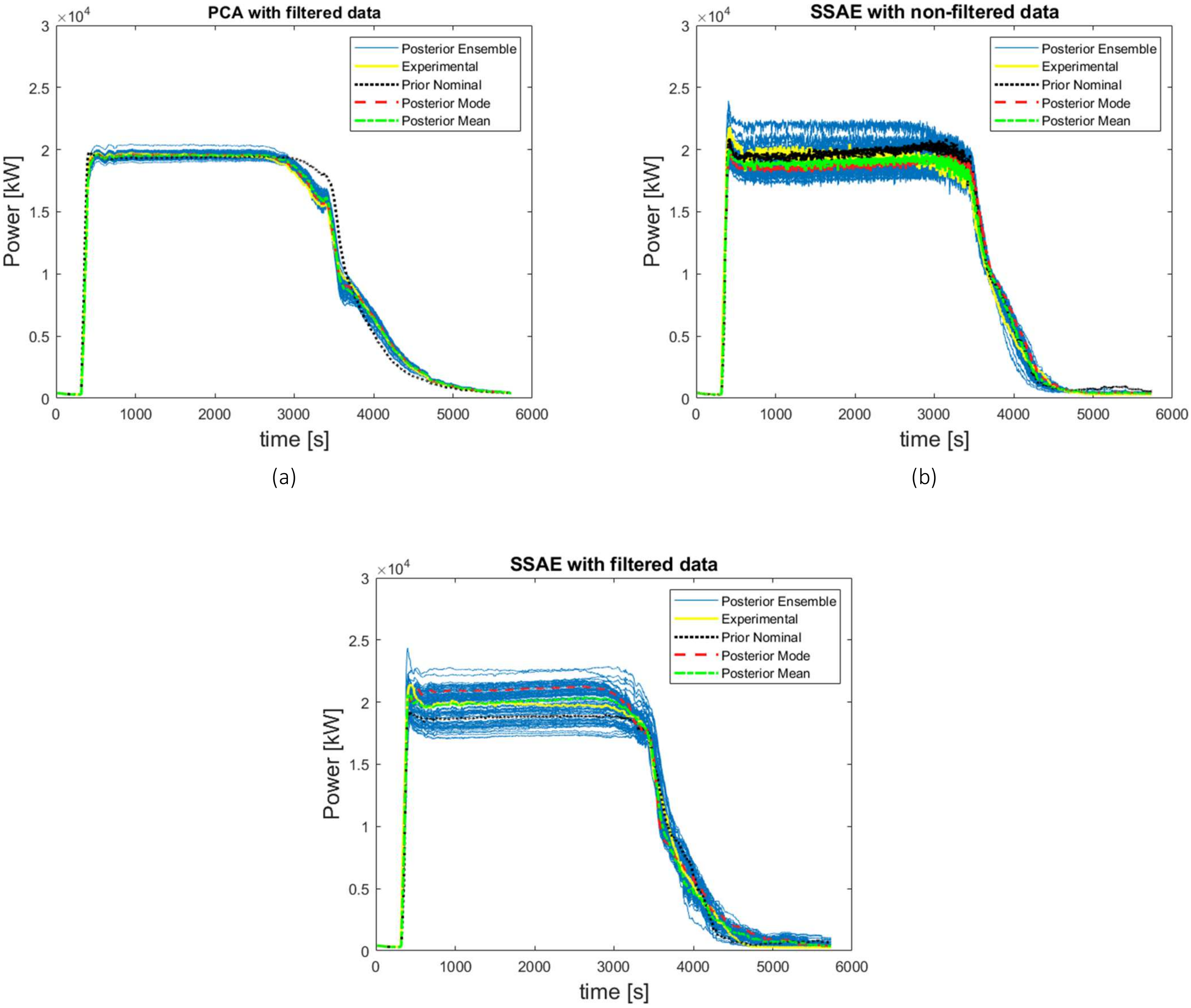
As a general result, the posterior PDFs obtained through the SSAE, with filtered and raw data, are wider than those obtained through PCA. In this view, the PCA-based approach, providing sharper posterior PDFs (i.e., characterized by smaller variances), seems to allow reducing epistemic uncertainty about calibration parameters more than the SSAE-based approach. However, it is difficult to assess and comment on the consistency of such posterior PDFs; in this regard, Section 5.3 proposes the propagation of such uncertainty to check the consistency of the results with respect to experimental data.

5.3 Forward Uncertainty Quantification: comparison of SSAE and PCA

To compare and assess the quality of the proposed IUQ approaches, we perform forward uncertainty propagation. In particular, (1) we simulate the posterior samples obtained from the MCMC after an appropriate thinning and (2) we compare the ensemble of time series obtained to the experimental data $\boldsymbol{y}^{\boldsymbol{E}}$ of Test 7-Part 2; this allows comparing the posterior PDFs obtained applying both PCA and SSAE. Both the RELAP5-3D model and the Kriging metamodels are used to propagate the posteriors' uncertainty. The Kriging-based forward uncertainty propagation is proposed in Section 5.3.1, whereas Section 5.3.3 shows the results obtained by propagating through the RELAP5-3D model.

5.3.1 Kriging metamodels

For each IUQ approach proposed, 100 posterior samples are simulated through the Kriging metamodels to obtain the predictions in the p^*-dimensional reduced space. Such predictions are, then, transformed into the p-dimensional space, adopting transformation matrix $\boldsymbol{\Phi}$ in the case of PCA and the SSAE decoders in the other case. In particular, Figure 9a, Figure 9b, Figure 9c compare the reconstructed experimental data $\breve{\boldsymbol{y}}^{\boldsymbol{E}}$ to the reconstructed Kriging predictions of 100 posterior samples, the prior nominal value, the posterior mode and the posterior mean value.



(a)

(b)

(c)

Figure 9. Reconstructed Kriging predictions of $\mathbf{100}$ posterior samples, the prior Nominal value, the posterior mode and the posterior mean value, compared with the reconstructed experimental data. The results obtained applying PCA with filtered data are reported in Figure 9a, whereas Figure 9b and Figure 9c report the results obtained applying the SSAE with non-filtered and filtered data, respectively.

In each case, the simulated posterior samples envelop $\breve{\boldsymbol{y}}^{\boldsymbol{E}}$. Moreover, in the case of PCA, the posterior's prediction ensemble (Figure 9a) shows smaller variance than that of the SSAE (Figure 9b and Figure 9c); this is in line with the narrower PDFs that characterize the posterior marginal KDE obtained applying PCA (Figure 8). In general, the reconstructed Kriging predictions of the posterior samples obtained adopting PCA reproduce $\breve{\boldsymbol{y}}^{\boldsymbol{E}}$ better than the reconstructed Kriging predictions of the posterior samples obtained adopting the SSAE with both filtered and raw data. Finally, we can notice that, when the SSAE is applied, data filtering does not bring a significant improvement in terms of agreement of the simulated posterior samples with $\breve{\boldsymbol{y}}^{\boldsymbol{E}}$. It is worth mentioning that the low number of simulated posterior samples (i.e., 100) is due to the fact that, as explained in Section 5.2.2, thinning has been performed on the MCMC samples to reduce autocorrelation, reducing the effective number of MCMC posterior samples to 1000 and 100, in PCA and SSAE cases, respectively.

5.3.2 Safety margin calculation

In BEPU methodologies, the results are expressed in terms of uncertainty ranges for the calculated Figure of Merit (FOM); this allows to compute safety margins with respect to safety threshold values. The current BEPU methods can be subdivided in: (1) probabilistic methods based on the propagation of input uncertainties (e.g., CSAU, GRS and ASTRUM), (2) deterministic methods based on the propagation of input uncertainties (e.g., AEAW and EDF-Framatome), and (3) methods based on the extrapolation of output uncertainty (e.g., UMAE). Following the GRS method [56], Wilk's formula [57] can be applied to determine the number of simulations N that guarantees the confidence level β and the probability content γ [58]. In the case of one-sided confidence level on one FOM, the Wilk's formula reduces to:

$$1-\gamma^N \geq \beta \tag{21}$$

This expression is valid for first-order Wilks' formula, that is the case in which the highest (lowest) outcome is within the upper (lower) 5% range with at least 95% confidence.

Following [35], the integral over the mission time $T_{mission} = 5736\ s$ (i.e., the duration of test 7 Part 2) of the HX exchanged power $P(t)$ is selected as FOM, since it is a relevant parameter for the current analysis:

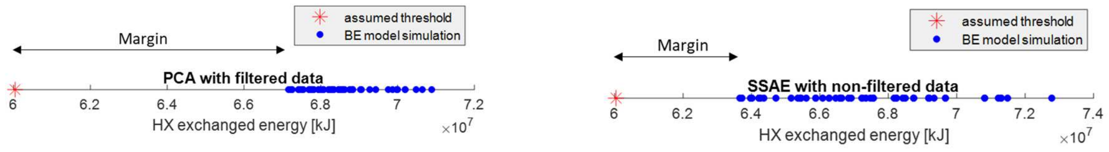
$$E = \int_0^{T_{mission}} P(t)dt \tag{22}$$

For demonstration purposes, in the three cases analyzed (i.e., PCA with filtered data and SSAE with both filtered and non-filtered data), the acceptance criterion for the selected FOM is set to $0.9E_{nominal}$, i.e., the system is considered failed if $E < 0.9\int_0^{T_{mission}} P_{nominal}(t)dt$, where $P_{nominal}$ is the RELAP5-3D simulated HX exchanged power using as input the prior nominal values. According to Wilks' formula [57], for each case, $N = 59$ RELAP5-3D simulations are carried out to calculate the one-sided statistical tolerance limit with $\beta = 95\%$ confidence level and $\gamma = 95\%$ probability content (see Eq. (21)). According to Wilks's formula, the lowest simulated value of E (defined as E_{lowest}) is within the lower 5% range with at least 95% confidence; thus, its margin to the assumed threshold value (i.e., $0.9E_{nominal} = 6.004 \times 10^7 kJ$) is computed as $M = E_{lowest} - 0.9E_{nominal}$. Table 9 reports the values of E_{lowest} and the respective margins M (with respect to the threshold value $0.9E_{nominal}$); the latter values are also graphically reported, along with the $N = 59$ computed values of E, in Figure 10a, Figure 10b and Figure 10c. It is important to remark that the uncertainty analysis carried out in this Section is performed only for demonstration purposes and involves only the propagation of epistemic uncertainty (i.e., the one characterized through the IUQ process).

It can be noticed that the safety margins computed for the SSAE are lower than those obtained for the PCA with filtered data; this is a direct consequence of the larger variance that characterizes the posterior PDFs found through the SSAE-based approach of IUQ. In this regard, if aleatory uncertainty is also considered, the margin with respect to the assumed threshold value can even be lower; thus, for a realistic calculation of the safety margin, it is recommended to properly characterize such aleatory uncertainty and repeat the safety margin calculation.

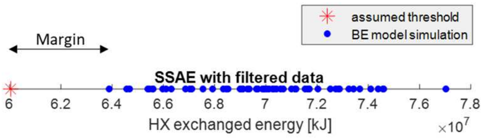
Table 9. The lowest simulated values of HX exchanged energy and the respective margins with respect to the threshold value $\mathbf{0.9}\boldsymbol{E_{nominal}}$.

	$\boldsymbol{E_{lowest}}$	$\boldsymbol{M}$
PCA with filtered data	$\mathbf{6.716 \times 10^7}\ \boldsymbol{kJ}$	$\mathbf{7.120 \times 10^6}\ \boldsymbol{kJ}$
SSAE with non-filtered data	$\mathbf{6.366 \times 10^7}\ \boldsymbol{kJ}$	$\mathbf{3.618 \times 10^6}\ \boldsymbol{kJ}$
SSAE with filtered data	$\mathbf{6.388 \times 10^7}\ \boldsymbol{kJ}$	$\mathbf{3.844 \times 10^6}\ \boldsymbol{kJ}$



(a)

(b)

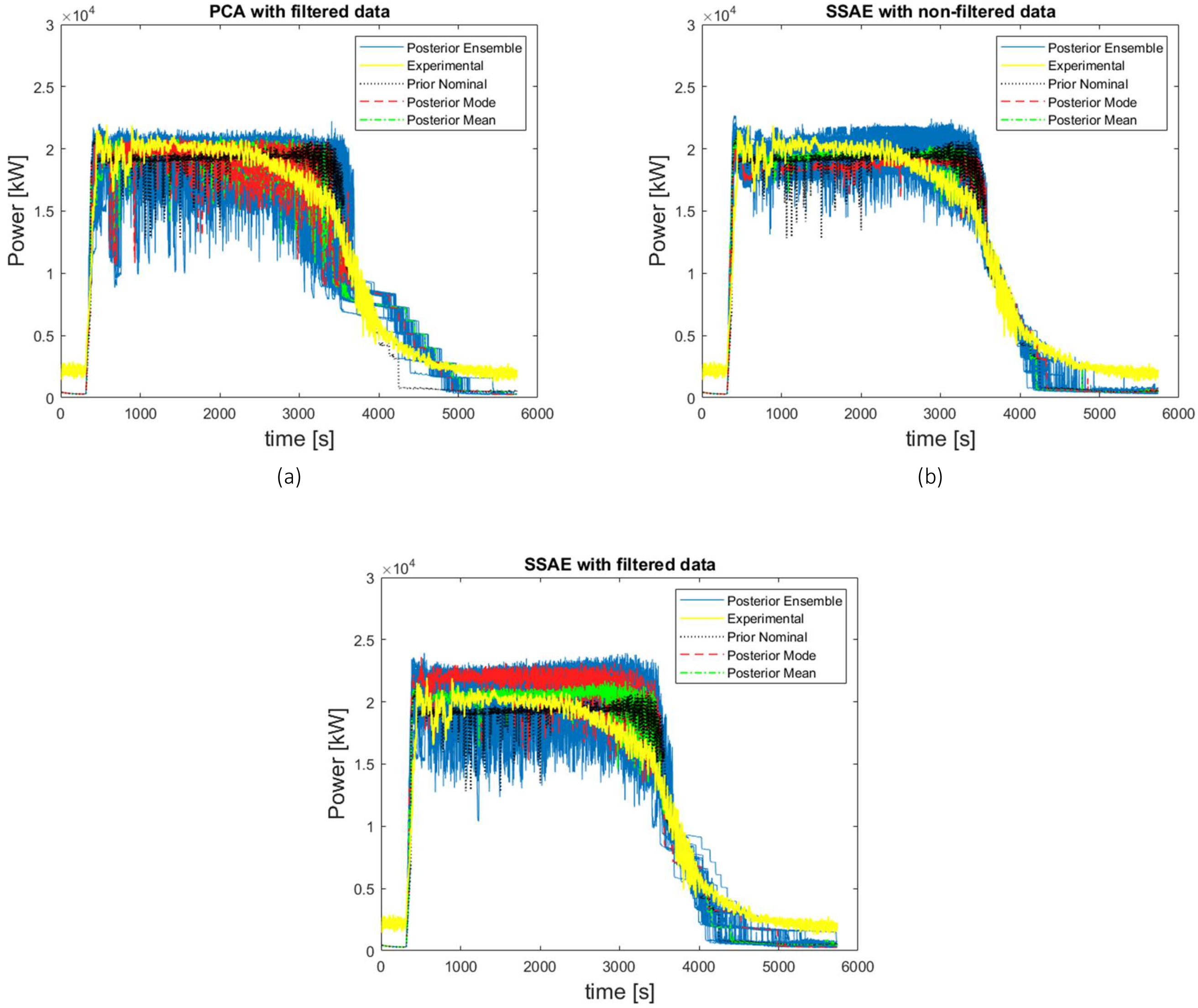


(c)

Figure 10. Safety margins with respect to the threshold value (i.e., $\mathbf{0.9}\boldsymbol{E_{nominal}}$) in the case of PCA with filtered data (Figure 10a), SSAE with non-filtered data (Figure 10b) and SSAE with filtered data (Figure 10c).

5.3.3 RELAP5-3D

The good agreement observed for the PCA-based approach (applied to filtered data), between $\breve{\boldsymbol{y}}^{\boldsymbol{E}}$ and the simulated posterior samples in the Kriging-based forward uncertainty propagation must be further investigated when the RELAP5-3D BE model is adopted. The $N = 59$ available RELAP5-3D simulations, carried out to calculate the safety margins as in Section 5.3.2, are used to assess and compare the PCA and the SSAE from 1) a qualitative point of view and 2) a quantitative point of view, by computing the Signal to Noise Ratio. More precisely, Figure 11a, Figure 11b and Figure 11c compare the Test 7-Part 2 experimental data with respect to the RELAP5-3D predictions of the following input parameters: $N = 59$ posterior samples, the prior nominal value, the posterior mode $\boldsymbol{\theta}_{\boldsymbol{posterior}}^{\boldsymbol{mode}}$ and posterior mean value $\boldsymbol{\theta}_{\boldsymbol{posterior}}^{\boldsymbol{mean}}$ (these two latter obtained from the thousands of MCMC samples after burn-in).



(a)

(b)

(c)

Figure 11. RELAP5-3D predictions of $\boldsymbol{N = 59}$ posterior samples compared to experimental data and the RELAP5-3D prediction of the prior nominal value, the posterior mode and posterior mean value. The results obtained applying PCA with filtered data are reported in Figure 11a, whereas Figure 11b and Figure 11c report the results obtained applying the SSAE with non-filtered and filtered data, respectively.

One can notice that, when PCA is applied to filtered data, the RELAP5-3D simulated posterior mode $\boldsymbol{y}^M(\boldsymbol{\theta}_{\boldsymbol{posterior,PCA}}^{\boldsymbol{mode}})$ and mean value $\boldsymbol{y}^M(\boldsymbol{\theta}_{\boldsymbol{posterior,PCA}}^{\boldsymbol{mean}})$ (and, in general, all the $N = 59$ simulated posterior samples) display a much wider noise with respect to the case in which the SSAE is applied to non-filtered data. Since the oscillations affecting the RELAP5-3D simulated HX power exchanged are higher in comparison with the experimental data, their signal-to-noise ratio can be considered a characteristic to evaluate their physical consistency: the lower the signal-to-noise ratio, the lower the physical consistency. In this regard, for the PCA and the SSAE with/without filter results shown in Figure 11, we compute the SNR (see Eq.(23)) of the $N = 59$ RELAP5-3D output time series: we model the (unknown) noise-free output $p \times N$ matrix $\boldsymbol{S}$ with a moving median filter (with a $50\ s$ sliding window to accommodate sudden power oscillations, i.e. $\sim 1\ MW$, that are assumed to be unrealistic) and we compute the $p \times N$ noise matrix $\boldsymbol{\xi}$ by subtracting $\boldsymbol{S}$ from the non-filtered output (i.e., $\xi_{ij} = y_j^{RELAP}(\boldsymbol{\theta}_i) - S_{ij}$ with $i = 1,2, \dots, N$ and $j = 1,2, \dots p$):

$$SNR = 20 \log_{10} \left(\frac{\sqrt{\sum_{i=1}^{N} \sum_{j=1}^{p} \left| S_{ij} \right|^2}}{\sqrt{\sum_{i=1}^{N} \sum_{j=1}^{p} \left| \xi_{ij} \right|^2}} \right) \tag{23}$$

For PCA and SSAE with/without filtering, the SNR values are equal to: $SNR_{PCA} = 23.04\ dB$, $SNR_{SSAE,noisy} = 31.12\ dB$ and $SNR_{SSAE,filtered} = 26.15\ dB$. Concerning this last point, the SSAE-based approach applied to raw data outperforms the PCA-based approach. Moreover, comparing Figure 11b and Figure 11c, we can notice that data-filtering does not provide any benefits both in terms of posterior mode accuracy (in predicting $\boldsymbol{y^E}$) and signal-to-noise ratio (characterizing the $N = 59$ simulated posterior samples), when the SSAE-based approach is applied. A possible reason is that filtered data may bias autoencoders and Kriging training. This may impair the metamodel generalization capability when fed with noisy data. This finally suggests that filtering should be avoided in favour of the use of raw data, whose information carried is significative and can be capitalized by the autoencoder in SSAE-based IUQ approaches.

6 Conclusions

In the present paper, we propose a novel approach for the uncertainty quantification of T-H model code input parameters in case of noisy (i.e., small signal-to-noise ratio) time-dependent outputs, that paves the way for a novel dimensionality reduction method capable of dealing with raw data in the context of Bayesian IUQ. The approach is developed within a Bayesian framework and adopts a SSAE-based dimensionality reduction technique to extract significant features from the simulated time series and involves implementing Kriging metamodels for the quick emulation of such features.

The effectiveness of the proposed approach has been firstly demonstrated considering (noisy) raw data simulated by a time-dependent RELAP5-3D model the PERSEO facility in relation to the power exchanged by the HX. The results show *i)* the capability of the adopted SSAE to reduce the input data dimensionality while preserving its most significant characteristics and *ii*) the ability of the proposed IUQ approach in dealing with (noisy) raw data. Moreover, the comparison of the proposed approach with a standard dimensionality reduction method (i.e., PCA), exhibit the inability of the latter in dealing with (noisy) raw data. This highlights the novel characteristic of the SSAE-based approach, which, in contrast to the standard dimensionality reduction methods, allows going through the IUQ without resorting to filtering techniques, which are based on expert judgment and can affect the IUQ results. Moreover, the PDFs obtained applying the SSAE-based approach to raw data, when propagated through the RELAP5-3D code, give power exchanged by the HX time series characterized by a higher signal-to-noise ratio than in the PCA-based approach. This can be considered an element to assess the physical consistency of the uncertainty propagated to the code output, which theoretically should not be affected by large noise/oscillations.

Also, the proposed approach has been applied to a filtered data set, still related with the RELAP5-3D model of the PERSEO facility. It can be concluded that performing data filtering before applying the SSAE does not bring any benefits in terms of metamodel accuracy and consistency of the propagated results with respect to experimental data. This is in line with the fact that SSAEs, being able to perform denoising, should not be affected by the noise. The comparison with the PCA-based approach (applied to filtered data) shows that *i*) PCA allows reducing epistemic uncertainty more than the SSAE-based approach since the former provides sharper posterior PDFs (i.e., characterized by minor variance) and *ii*) the MCMC sampling is computationally more expensive for SSAE than for PCA.

Future research lies on the possibility of exploring: 1) more powerful tuning approaches to optimize the SSAE architecture (e.g., extensive grid search and evolutionary optimization); 2) new approaches of uncertainty propagation through DNNs (e.g., the Monte Carlo sampling, the entire-DNN unscented transform and the piecewise exponential approximation of the activation function) taking, also, into account their computational cost; 3) new approaches, such as multivariate Kriging metamodels [26], to take into account dependencies among the output components. In fact, another limitation that should be further investigated is the effect of building a separate independent metamodel for each feature extracted, i.e., assuming the features to be independent.

Appendix

Appendix A. Kriging Metamodeling

A Metamodel (MM) is a functional approximation of the input/output relations of the original model that is faster to be evaluated. It is constructed from a set of input parameters and corresponding model responses obtained by running the original time-demanding model. In this work, we adopt the Kriging metamodel because it provides a direct estimation of the metamodel uncertainty $\boldsymbol{\Sigma}_{MM}$ of equation (5).

A.1 Kriging Theory

Let us consider a general mathematical model $z = z(\boldsymbol{\theta})$, where z is a scalar and $\boldsymbol{\theta} = [\theta_1, \dots, \theta_d]$ is a d-dimensional vector. Assume that $\boldsymbol{\Theta} = \left[\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(m)}\right]^T$ is the Design Of Experiments (DOE) and $\boldsymbol{z} = [z_1, \dots, z_m]^T$ are the corresponding m model responses. Kriging is a method of interpolation which assumes that the model output $z(\boldsymbol{\theta})$ is a realization of a Gaussian process indexed by $\boldsymbol{\theta} \in \boldsymbol{D_\theta} \subset \mathbb{R}^{\boldsymbol{d}}$. The mathematical form of a Kriging is given by:

$$z(\boldsymbol{\theta}) = \sum_{j=1}^{n} \beta_j f_j(\boldsymbol{\theta}) + G(\boldsymbol{\theta}) = \boldsymbol{\beta^T f}(\boldsymbol{\theta}) + G(\boldsymbol{\theta}) \tag{A.1}$$

The first term in Eq. (A.1), $\boldsymbol{\beta^T f}(\boldsymbol{\theta})$, is the *mean* value of the Gaussian process, also called the "trend"; it is given by n arbitrary function $[f_1(\boldsymbol{\theta}), \dots, f_n(\boldsymbol{\theta})]$ and the related coefficient $[\beta_1, \dots, \beta_n]$. The second term, $G(\boldsymbol{\theta})$, is a Gaussian process with zero mean and covariance:

$$Cov\left[G\left(\boldsymbol{\theta}^{(i)}\right), G\left(\boldsymbol{\theta}^{(j)}\right)\right] = \sigma^2 R\left(\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(j)}\right) \tag{A.2}$$

Where $\sigma^2$is the *process variance* and $R(\cdot,\cdot)$ is the *correlation function* (also called *correlation kernel*) defined for any two points in the input domain $\boldsymbol{D_\theta}$.

$R(\cdot,\cdot)$ is a function of the distance $h(\cdot,\cdot)$:

$$R\left(\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(j)}\right) = R\left(h(\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(j)})\right) \tag{A.3}$$

According to [19], $h(\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(j)})$ has the following expression:

$$h\left(\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(j)}\right) = \left[\sum_{k=1}^{d} \left(\frac{\theta_k^{(i)} - \theta_k^{(j)}}{\omega_k}\right)^2\right]^{0.5} \tag{A.4}$$

Where the parameters $\boldsymbol{\omega} = [\omega_1, \dots, \omega_d]$ are called *length scale parameters*. Table 10 and Table 11 show, respectively, some common correlation kernels and trend types that are implemented and compared in this work to find the best Kriging options.

Table 10. Common correlation kernels .

Correlation Kernel	Expression
Exponential	$\boldsymbol{R}\left(\boldsymbol{\theta}^{(i)},\boldsymbol{\theta}^{(j)}\right)=\mathbf{exp}\left[-\boldsymbol{h}\left(\boldsymbol{\theta}^{(i)},\boldsymbol{\theta}^{(j)}\right)\right]$
Gaussian	$\boldsymbol{R}\left(\boldsymbol{\theta}^{(i)},\boldsymbol{\theta}^{(j)}\right)=\mathbf{exp}\left[-\frac{\mathbf{1}}{\mathbf{2}}\boldsymbol{h}^{2}\left(\boldsymbol{\theta}^{(i)},\boldsymbol{\theta}^{(j)}\right)\right]$
Matérn-3/2	$\boldsymbol{R}\left(\boldsymbol{\theta}^{(i)},\boldsymbol{\theta}^{(j)}\right)=\left(\mathbf{1}+\sqrt{\mathbf{3}}\cdot\boldsymbol{h}\left(\boldsymbol{\theta}^{(i)},\boldsymbol{\theta}^{(j)}\right)\right)\mathbf{exp}\left[-\sqrt{\mathbf{3}}\cdot\boldsymbol{h}\left(\boldsymbol{\theta}^{(i)},\boldsymbol{\theta}^{(j)}\right)\right]$
Matérn-5/2	$\boldsymbol{R}\left(\boldsymbol{\theta}^{(i)},\boldsymbol{\theta}^{(j)}\right)=\left(\mathbf{1}+\sqrt{\mathbf{5}}\cdot\boldsymbol{h}\left(\boldsymbol{\theta}^{(i)},\boldsymbol{\theta}^{(j)}\right)+\frac{\mathbf{5}}{3}\cdot\boldsymbol{h}^{2}\left(\boldsymbol{\theta}^{(i)},\boldsymbol{\theta}^{(j)}\right)\right)\mathbf{exp}\left[-\sqrt{\mathbf{5}}\cdot\boldsymbol{h}\left(\boldsymbol{\theta}^{(i)},\boldsymbol{\theta}^{(j)}\right)\right]$

Table 11. Common Trend Types

Trend type	Expression
Constant (ordinary Kriging)	$\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{f}(\boldsymbol{\theta})=\boldsymbol{\beta}_{\mathbf{0}}$
Linear	$\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{f}(\boldsymbol{\theta})=\boldsymbol{\beta}_{\mathbf{0}}+\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{d}}\boldsymbol{\beta}_{\boldsymbol{i}}\boldsymbol{\theta}_{\boldsymbol{i}}$
Quadratic	$\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{f}(\boldsymbol{\theta})=\boldsymbol{\beta}_{\mathbf{0}}+\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{d}}\boldsymbol{\beta}_{\boldsymbol{i}}\boldsymbol{\theta}_{\boldsymbol{i}}+\sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{d}}\sum_{\boldsymbol{j}=\mathbf{1}}^{\boldsymbol{d}}\boldsymbol{\beta}_{\boldsymbol{ij}}\boldsymbol{\theta}_{\boldsymbol{i}}\boldsymbol{\theta}_{\boldsymbol{j}}$

In line with the Kriging definition, $\boldsymbol{z}$ and a generic model prediction $z(\boldsymbol{\theta})$ at the untried input location $\boldsymbol{\theta}$ are jointly distributed according to a multivariate Gaussian distribution:

$$\begin{Bmatrix} z(\boldsymbol{\theta}) \\ \boldsymbol{z} \end{Bmatrix} \sim N\left(\begin{Bmatrix} \boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{f}(\boldsymbol{\theta}) \\ \boldsymbol{F\beta} \end{Bmatrix}, \sigma^2\begin{Bmatrix} 1 & \boldsymbol{r}^{\boldsymbol{T}}(\boldsymbol{\theta}) \\ \boldsymbol{r}(\boldsymbol{\theta}) & \boldsymbol{R} \end{Bmatrix}\right) \tag{A.5}$$

Where $\boldsymbol{F}$ is the matrix of regression function $\boldsymbol{f}(\boldsymbol{\theta})$ evaluated at the m design points (i.e., $F_{ij}=f_j(\boldsymbol{\theta}_{\boldsymbol{i}})$, $i=1,\dots,m$; $j=1,\dots,n$); $\boldsymbol{r}(\boldsymbol{\theta})$ is the vector of correlations between $\boldsymbol{\theta}$ and the DOE points (i.e., $r_i=R\left(\boldsymbol{\theta},\boldsymbol{\theta}^{(\boldsymbol{i})}\right)$ $i=1,\dots,m$); $\boldsymbol{R}$ is the correlation matrix (i.e., $R_{ij}=R\left(\boldsymbol{\theta}^{(j)},\boldsymbol{\theta}^{(i)}\right)$ $i,j=1,\dots,m$).

The Kriging prediction (i.e., the mean value) $\hat{z}(\boldsymbol{\theta})$ and the variance $\sigma^2_{z(\boldsymbol{\theta})}$ of the Gaussian random variable $z(\boldsymbol{\theta})$ are given by [59]:

$$\hat{z}(\boldsymbol{\theta})=\boldsymbol{f}(\boldsymbol{\theta})^T\widehat{\boldsymbol{\beta}}+\boldsymbol{r}(\boldsymbol{\theta})^T\boldsymbol{R}^{-1}(\boldsymbol{z}-\boldsymbol{F}\widehat{\boldsymbol{\beta}}) \tag{A.6}$$

$$\sigma^2_{z(\boldsymbol{\theta})}=\sigma^2\left(1-\boldsymbol{r}^T(\boldsymbol{\theta})\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{\theta})+\left(\boldsymbol{F}^{\boldsymbol{T}}\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{\theta})-\boldsymbol{f}(\boldsymbol{\theta})\right)^{\boldsymbol{T}}(\boldsymbol{F}^{\boldsymbol{T}}\boldsymbol{R}^{-1}\boldsymbol{F})^{-1}\left(\boldsymbol{F}^{\boldsymbol{T}}\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{\theta})-\boldsymbol{f}(\boldsymbol{\theta})\right)\right) \tag{A.7}$$

where $\widehat{\boldsymbol{\beta}}=\left(\boldsymbol{F}^{\boldsymbol{T}}\boldsymbol{R}^{-\mathbf{1}}\boldsymbol{F}\right)^{-1}\boldsymbol{F}^{\boldsymbol{T}}\boldsymbol{R}^{-\mathbf{1}}\boldsymbol{z}$ is the least square estimate of the regression coefficients.

A crucial consequence of the Gaussian assumption is that:

$$z(\boldsymbol{\theta})\sim N\left(\hat{z}(\boldsymbol{\theta}),\sigma^2_{z(\boldsymbol{\theta})}\right). \tag{A.8}$$

A property of the Kriging predictor is that it always interpolates the design sites. The Mean square error of the prediction (i.e., $\sigma^2_{z(\boldsymbol{\theta}^*)}$) collapses to zero as the untried point $\boldsymbol{\theta}^*$ gets close to the design sites and increases when $\boldsymbol{\theta}^*$ moves away from the design sites. Intuitively, a richer DOE will give a Kriging with a better prediction capability. However, increasing the DOE size is generally computationally intensive; therefore, selecting design sites through an efficient DOE approach is a crucial issue to be addressed. Latin hypercube sampling is one of the most popular DOE techniques [60], and it is adopted in this work.

To formulate the general theory of Kriging, we introduced $(n + d + 1)$ hyperparameters (i.e., n regression coefficients $\boldsymbol{\beta}$, d length scale parameters $\boldsymbol{\omega}$, one process variance σ^2) that are unknown and thus must be estimated. Estimation methods such as M*aximum likelihood estimation* (ML) or *Cross-Validation estimation* (CV) are usually applied to estimate them. In this work, we adopt a software package, UQLab [19], to implement the Kriging metamodeling. Further details regarding the comparison between ML and CV can be found in [61].

A.2 Kriging performance indicators

The quality assessment for a Kriging model is usually performed considering the accuracy in reproducing the original model output at unobserved locations [62]. This is typically done by measuring the error in Kriging's prediction using a *validation dataset* (i.e., a set of model simulations different from $\boldsymbol{\Theta}$). However, this approach requires a considerable number of additional simulations that likely are computationally intensive. A computationally cheaper alternatives, namely the *cross-validation* (CV) can be used to estimate the metamodel quality without any additional simulation beyond those used to train the Kriging [62]. According to the Leave One Out Cross Validation (LOOCV) criterion, the metamodel is trained leaving one simulation out from the training dataset, and then it is used to predict the remaining simulation outcomes. Following this criterion the normalized LOOCV error ϵ_{LOOCV} is computed as:

$$\epsilon_{LOOCV} = \frac{\frac{1}{m}\sum_{i=1}^{m}\left(z(\boldsymbol{\theta}^{(i)}) - \hat{z}_{(-i)}(\boldsymbol{\theta}^{(i)})\right)^2}{\frac{1}{m}\sum_{i=1}^{m}(z(\boldsymbol{\theta}^{(i)}) - \overline{z})^2} \tag{A.9}$$

where m is the training dataset size, $\hat{z}_{(-i)}(\boldsymbol{\theta}^{(i)})$ is the prediction of the Kriging that is trained using all the points of $\boldsymbol{\Theta}$, except $\boldsymbol{\theta}^{(i)}$, $\overline{z} = \frac{1}{m}\sum_{i=1}^{m} z(\boldsymbol{\theta}^{(i)})$. It should be noticed that in expression (A.9), the LOOCV error is normalized with respect to the training output sample variance (i.e., $\frac{1}{m}\sum_{i=1}^{m}\left(z(\boldsymbol{\theta}^{(i)}) - \overline{z}\right)^2$).

Appendix B. PCA-based Inverse Uncertainty Quantification with filtered data

B.1 Principal Component Analysis

PCA is a space transformation from a high dimensional space to a lower-dimensional space such that transformed variables are uncorrelated and retain as much as possible of the variation present in the data set [30]. Singular Value Decomposition (SVD), as well as the eigendecomposition of the covariance matrix of the data, can be adopted to perform PCA; the former is proposed below. Let $\boldsymbol{Y} = [\boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(m)}]$ be the $p \times m$ *data matrix* containing the m, p-dimensional, model output responses (also called *observations*) (where $\boldsymbol{y}^{(i)} = [y_1{}^{(i)}, \dots, y_p{}^{(i)}]^T$). The centered data matrix $\boldsymbol{Y}_{centered}$ is defined as:

$$\boldsymbol{Y}_{centered} = [(\boldsymbol{y}^{(1)} - \overline{\boldsymbol{\mu}}_{\boldsymbol{Y}}), \dots, (\boldsymbol{y}^{(m)} - \overline{\boldsymbol{\mu}}_{\boldsymbol{Y}})] \tag{B.1}$$

Where

$$\overline{\boldsymbol{\mu}}_{\boldsymbol{Y}} = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{y}^{(i)} \tag{B.2}$$

The SVD decomposition of $\boldsymbol{Y}_{centered}$ is [63]:

$$\boldsymbol{Y}_{centered} = U\Lambda V^T \tag{B.3}$$

Where U is a $p \times p$ unitary matrix whose columns $\boldsymbol{u}_i$ are called *left singular vectors*, Λ is a $p \times m$ diagonal matrix whose entries $\sqrt{\lambda_j}$ are called *singular values*, V^T is the conjugate transpose of V that is a $m \times m$ unitary matrix. The columns of V^T are called *right singular vectors*. There is a direct relation between PCA and SVD [63]:

- the entries of Λ (i.e., $\sqrt{\lambda_j}$) are the square roots of the eigenvalues (arranged in descending order) of $\boldsymbol{Y}_{centered}\boldsymbol{Y}^{\boldsymbol{T}}_{centered}$, thus they are proportional to the covariance matrix eigenvalues;
- $\boldsymbol{u}_j$ are the *principal components vectors* (PCs).

To determine the dimension of the principal subspace p^* one finds the smallest p^*such that the cumulative percentage of variation explained $\left(i.e., \frac{\sum_{j=1}^{p^*}\lambda_j}{\sum_{j=1}^{p}\lambda_j} \cdot 100\right)$ is at least equal to a threshold value (usually chosen between 95% and 99%) [20]. The first p^* PCs $\boldsymbol{u}_j$ form the $p^* \times p$ transformation matrix Φ:

$$\Phi = \begin{bmatrix} \boldsymbol{u}_1^{\boldsymbol{T}} \\ \vdots \\ \boldsymbol{u}_{p^*}^{\boldsymbol{T}} \end{bmatrix} \tag{B.4}$$

The transformation of the data matrix into the feature space is given by:

$$\boldsymbol{Z} = \Phi(\boldsymbol{Y} - \overline{\boldsymbol{\mu}}_{\boldsymbol{Y}}) = \left[\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\right] = \begin{bmatrix} z_1^{(1)} & \cdots & z_1^{(m)} \\ \vdots & \ddots & \vdots \\ z_{p^*}^{(1)} & \cdots & z_{p^*}^{(m)} \end{bmatrix} \tag{B.5}$$

where $\boldsymbol{Z}$ is a $p^* \times m$ matrix whose rows are called *PC scores* and columns represent the output high dimensional data projected into the features space. To antitransform a generic vector $\boldsymbol{z}$ from the features space to the original space:

$$\boldsymbol{y} = \overline{\boldsymbol{\mu}}_{\boldsymbol{Y}} + \Phi^T \boldsymbol{z} \tag{B.6}$$

B.2 PCA-based IUQ with filtered data

Given the limitations of the PCA to cope with raw data (i.e., ϵ_{LOOCV} close to 1.0 for most of the features), a possible solution is to filter the HX power exchanged predicted by RELAP5-3D (noisy) raw time series collected in $\boldsymbol{Y}$ before applying PCA and Kriging metamodels. Although this allows removing the part of output variability due to numerical oscillations, data filtering gives rise to a nontrivial issue, i.e., choosing a proper filtering technique (that should consider the smallest timescale on which physical phenomena take place during the transient). Moreover, although filtering raw data affected by numerical oscillation may appear reasonable, (1) justifying the choice of a particular filtering method rather than another is a nontrivial task, and (2) the IUQ results may be affected by the selection of the specific filtering method adopted. Thus, by applying a moving median filter (i.e.,"movmedian" MATLAB function) on the columns of $\boldsymbol{Y}$, a smaller number of PCs (i.e., $p^* = 4$, rather than 31) is able explain the same percentage of variance of the dataset (i.e., 95%) [35]. We compute R_{error} and ϵ_{LOOCV} through (9) and (14), respectively, and we find $R_{error} = 2.1308 \times 10^5 kW$, whereas the four ϵ_{LOOCV} remain below 0.2, as shown in [35].

Since for the proposed case study $N_{exp} = 1$, the posterior formulations in case of PCA reduces to [35]:

$$p(\boldsymbol{\theta}|\boldsymbol{z}^{\boldsymbol{E}}) \propto p(\boldsymbol{\theta}) \cdot \frac{1}{\left(\sqrt{2\pi}\right)^{p^*}\sqrt{|\boldsymbol{\Sigma}|}} \exp\left[-\frac{1}{2}[\boldsymbol{z}^E - \hat{\boldsymbol{z}}^{\boldsymbol{MM}}(\boldsymbol{\theta})]^T \left(\boldsymbol{\Sigma}_{exp} + \boldsymbol{\Sigma}_{\boldsymbol{Kriging}}\right)^{-1} [\boldsymbol{z}^E - \hat{\boldsymbol{z}}^{\boldsymbol{MM}}(\boldsymbol{\theta})]\right] \tag{B.7}$$

where $\hat{\boldsymbol{z}}^{\boldsymbol{MM}}(\boldsymbol{\theta})$ is the mean value of the Kriging prediction and $\boldsymbol{\Sigma}_{exp} = \Phi \boldsymbol{I} \sigma_{exp}^2 \Phi^T$ is the experimental uncertainty covariance matrix in the p^*-dimensional reduced space. In the present work, σ_{exp} is set to $500\, kW$ [53]. $\boldsymbol{\Sigma}_{\boldsymbol{Kriging}}$ is the covariance matrix associated with the Kriging prediction uncertainty (see (17)). Eight parallel MCMC chains with $5 \cdot 10^5$ iterations are run by applying an adaptive Metropolis algorithm [64] implemented through the UQLab MATLAB package [65]. It took almost 2.5 hours to compute the posterior PDFs on an Intel Core i7-7500U processor. We post-process the samples discarding the first half of each chain for burn-in. The MCMC simulation convergence is examined through the approach proposed in [13]. The marginal Kernel Density Estimation (KDE) [66] of the posterior PDFs obtained through

PCA with filtered data is shown in Figure 8. The summary statistics of the posterior distribution (i.e., mean values, modes, 5^{th} and 95^{th} percentiles) are reported in Table 12. The results reported in this Section concerning the application of PCA on filtered data are elaborated from [35].

Table 12. Posterior summaries (PCA with filtered data).

$\boldsymbol{\theta_i}$	Parameter	Mean value	Mode	5th percentile	95th percentile
$\boldsymbol{\theta_1}$	Inner_FF	0.90	0.90	0.86	0.94
$\boldsymbol{\theta_2}$	Outer_FF	1.36	1.40	1.28	1.43
$\boldsymbol{\theta_3}$	K_injector	1.41	1.49	1.25	1.49
$\boldsymbol{\theta_4}$	K_sum_steam	0.96	0.88	0.56	1.40
$\boldsymbol{\theta_5}$	K_sum_condensate	0.99	1.06	0.55	1.45
$\boldsymbol{\theta_6}$	K_ diaphragm	0.96	1.30	0.54	1.44
$\boldsymbol{\theta_7}$	k_rockwool	1.21	1.04	1.02	1.45
$\boldsymbol{\theta_8}$	A_effective	1.43	1.48	1.33	1.50

.

Appendix C. Bayesian Inference (MCMC sampling)

The challenging objective of this Section is to formulate the likelihood $p\left(\boldsymbol{z^E}\middle|\boldsymbol{\theta}\right)$ in the case of non-linear dimensionality reduction of the output (e.g., SSAE). This problem can be tackled by propagating the uncertainty of $\boldsymbol{y^E}\boldsymbol{\sim N}(\boldsymbol{y}(\boldsymbol{\theta}),\boldsymbol{I}\sigma_{exp}^2)$ through the SSAE's decoder. Monte Carlo (MC) sampling and the Unscented Transform [67] have been employed in past works to propagate the uncertainty through Deep Neural Networks; that is, given a multivariate distribution of the input layer (e.g., $\boldsymbol{y^E}\boldsymbol{\sim N}(\boldsymbol{y}(\boldsymbol{\theta}),\boldsymbol{I}\sigma_{exp}^2)$, the mean vector $\boldsymbol{z}_{(L)}$ and the covariance matrix $\boldsymbol{\Sigma}_{(L)}$ of the output layer L are estimated [67,68]; however, these methods (particularly the MC sampling) can be computationally expensive [67,69]. Following the approach proposed in [69], in this work, we use Extended Kalman Filtering (EKF) [49] to propagate $\boldsymbol{y^E}\boldsymbol{\sim N}(\boldsymbol{y}(\boldsymbol{\theta}),\boldsymbol{I}\sigma_{exp}^2)$ through the SSAE encoder in order to derive an expression for the likelihood $p\left(\boldsymbol{z^E}\middle|\boldsymbol{\theta}\right)$. EKF is an algorithm that is used to estimate the state of non-linear discrete-time dynamic systems when the system state cannot be directly measured. In the EKF, the system's state at time-step l is treated as an uncertain quantity characterized by a mean vector $\boldsymbol{z}_{(l)}$ and a covariance matrix $\boldsymbol{\Sigma}_{(l)}$. The EKF algorithm consists of two steps: the prediction step and the update step. In the prediction step the system's state $\boldsymbol{z}_{(l)}$ is predicted along with its error covariance $\boldsymbol{\Sigma}_{(\boldsymbol{l})}$ starting from (1) the *process noise*, (2) the *control input* and (3) the *previous step's state*. In the updating step, the prior estimates computed in the prediction step are updated (through indirect measurement of the system state) to find the posterior estimate of the state and and its error covariance. For further details about Kalman filtering refer to [49].

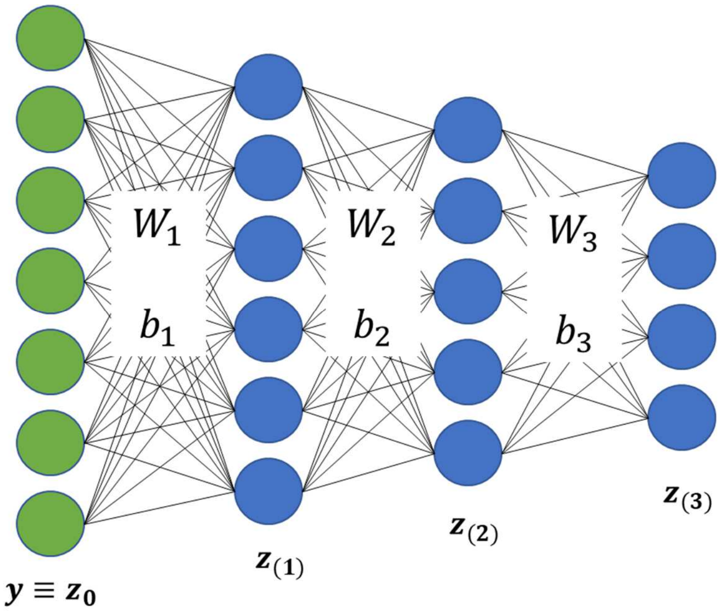

W_1
b_1
W_2
b_2
W_3
b_3
$\boldsymbol{y} \equiv \boldsymbol{z_0}$
$\boldsymbol{z}_{(\mathbf{1})}$
$\boldsymbol{z}_{(\mathbf{2})}$
$\boldsymbol{z}_{(\mathbf{3})}$

Figure 12. Example of DNN where $\boldsymbol{L} = \mathbf{3}$.

In the context of EKF applied for the uncertainty propagation in Deep Neural Networks, the concept of “system” does not refer to any real physical system; in fact, we treat the layers of the SSAE encoders as the states of a fictitious unforced (i.e., without a control input) system at different time-steps (i.e., the input layer of the SSAE $\boldsymbol{z}_{(0)}$ represents the system state at time $l = 0$, the second layer $\boldsymbol{z}_{(1)}$ represents the systems state at time $l = 1$ et cetera). Moreover, only the prediction step of the EKF algorithm is applied since there is no real physical system that allows measurements (used in the update step). Let us assume that the SSAE encoder is composed of $L + 1$ layers and that $\boldsymbol{z}_{(l)}$ and $\boldsymbol{\Sigma}_{(l)}$ are, respectively, the vector representing the state estimate (i.e., the mean value) and the covariance matrix of the layer l, such that:

$$\Sigma_{(l),j,k} = cov(z_{(l),j}, z_{(l),k}) \tag{C.1}$$

The system evolves from the state (read: layer) $l - 1$ to the state l through a non-linear transformation:

$$\boldsymbol{z}_{(l)} = \sigma(\boldsymbol{W}_{(l)}\boldsymbol{z}_{(l-1)} + \boldsymbol{b}_{(l)}) \tag{C.2}$$

where $\sigma(\cdot) = \frac{e^{(\cdot)}}{e^{(\cdot)}+1}$ is the sigmoid activation function of the SSAE, $\boldsymbol{W}_{(l)}$ is the SSAE weight matrix between layer $l - 1$ and layer l, $\boldsymbol{b}_{(l)}$ is the bias vector for layer l. Using the equation of the EKF’s prediction step, we can find the state estimate $\boldsymbol{z}_{(l)}$ and the covariance matrix $\boldsymbol{\Sigma}_{(l)}$ for each layer:

$$\boldsymbol{z}_{(l)} = \sigma\left(\boldsymbol{W}_{(l)}\boldsymbol{z}_{(l-1)} + \boldsymbol{b}_{(l)}\right) \tag{C.3}$$

$$\boldsymbol{\Sigma}_{(l)} = \boldsymbol{F}_{(l)}\boldsymbol{\Sigma}_{(l-1)}\boldsymbol{F}_{(l)} + \boldsymbol{Q}_{(l)} \tag{C.4}$$

where $\boldsymbol{F_l} = \nabla_{\boldsymbol{z}_{(l-1)}}\boldsymbol{z_l}$ is the Jacobian matrix and $\boldsymbol{Q_l}$ is the process noise covariance matrix that takes into account the inherent error introduced by the dimensionality reduction itself (i.e., the error related to the fact that the SSAE is not a perfect model). In this work, under the hypothesis that the SSAE is a perfect model, $\boldsymbol{Q_l}$ is neglected. If a sigmoid activation function is adopted, It can be shown that:

$$F_{(l),j,k} = \sigma\left(\sum_r W_{(l),j,r} \cdot z_{(l-1),r} + \boldsymbol{b}_{(l),j}\right)\left(1 - \sigma\left(\sum_r W_{(l),j,r} \cdot z_{(l-1),r} + b_{(l),j}\right)\right) \cdot W_{(l),j,k} \tag{C.5}$$

where $F_{(l),j,k}$ and $W_{(l),j,k}$ represent, respectively, the element at the j^{th} row and k^{th} column of the $\boldsymbol{F}_{(l)}$ and $\boldsymbol{W}_{(l)}$ matrices; $\boldsymbol{z}_{(l-1),r}$ is the r^{th} entry of $\boldsymbol{z}_{(l-1)}$. Iteratively applying equations (C.3) and (C.4), one can propagate $\boldsymbol{y^E} \sim \boldsymbol{N}(\boldsymbol{y}(\boldsymbol{\theta}), \boldsymbol{I}\sigma^2_{exp})$ through the SSAE encoder in order to derive the distribution of $\boldsymbol{z^E}$ that has mean value $\boldsymbol{z}_{(L)}$ and the covariance matrix $\boldsymbol{\Sigma}_{(L)}$. It should be noted that, since the sigmoid function is non-linear, the Jacobian matrices $\boldsymbol{F}_{(l)}$, and, in turn $\boldsymbol{\Sigma}_{(L)}$, both depend on the input vector $\boldsymbol{z}_{(0)} \equiv \boldsymbol{y}(\boldsymbol{\theta})$ that is a priori unknown since the Kriging metamodels predict directly $\hat{\boldsymbol{z}}^{\boldsymbol{MM}}_{\boldsymbol{(L)}}(\boldsymbol{\theta})$. To address this problem, we reconstruct $\hat{\boldsymbol{z}}^{\boldsymbol{MM}}_{\boldsymbol{(L)}}(\boldsymbol{\theta})$ (through the SSAE decoder) obtaining $\breve{\boldsymbol{y}}^K(\boldsymbol{\theta})$ that is eventually used to derive $\boldsymbol{\Sigma}_{(L)}$ through the procedure shown above. This procedure, unlike in the case of PCA, is repeated for each iteration of the MCMC algorithm because $\boldsymbol{\Sigma}_{(L)}$ depends on $\boldsymbol{\theta}$. It is unlikely to assume that $p(\boldsymbol{z^E}|\boldsymbol{\theta})$ is exactly multivariate Gaussian because of the non-linear transformations introduced by the sigmoid activation function; however, it still can be approximated by Gaussian distribution:

$$p(\boldsymbol{z^E}|\boldsymbol{\theta}) = \boldsymbol{N}(\boldsymbol{z^E}|\boldsymbol{z}_{(L)}, \boldsymbol{\Sigma}_{(L)}) \tag{C.6}$$

Considering the observation reported above; for each iteration of the MCMC sampling, we propose the following algorithm to determine and compute the analytical expression of the likelihood:

1. Compute the Kriging prediction $\hat{\boldsymbol{z}}^{\boldsymbol{MM}}_{\boldsymbol{(L)}}(\boldsymbol{\theta})$ in the p^*-dimensional features space;
2. reconstruct the Kriging prediction through the SSAE decoder in order to obtain $\breve{\boldsymbol{y}}^K(\boldsymbol{\theta})$;
3. impose $\boldsymbol{z}_{(0)} = \breve{\boldsymbol{y}}^K(\boldsymbol{\theta})$ and $\boldsymbol{\Sigma}_{(0)} = \boldsymbol{I}\sigma^2_{exp}$, then compute $\boldsymbol{\Sigma}_{(L)}$ applying the EKF;
4. assume the likelihood $p(\boldsymbol{z^E}|\boldsymbol{\theta})$ to be Gaussian distributed:

$$\boldsymbol{z^E} \sim N(\boldsymbol{z^E}|\boldsymbol{z}^{\boldsymbol{MM}}_{\boldsymbol{(L)}}(\boldsymbol{\theta}), \boldsymbol{\Sigma}_{(L)}(\boldsymbol{\theta})) = \boldsymbol{z}^{\boldsymbol{MM}}_{\boldsymbol{(L)}}(\boldsymbol{\theta}) + N(\boldsymbol{0}\,, \boldsymbol{\Sigma}_{(L)}(\boldsymbol{\theta})) \tag{C.7}$$

according to the Kriging theory:

$$\boldsymbol{z}^{\boldsymbol{MM}}_{\boldsymbol{(L)}}(\boldsymbol{\theta}) \sim N(\hat{\boldsymbol{z}}^{\boldsymbol{MM}}_{\boldsymbol{(L)}}(\boldsymbol{\theta}), \boldsymbol{\Sigma}_{Kriging}(\boldsymbol{\theta})) = \hat{\boldsymbol{z}}^{\boldsymbol{MM}}_{\boldsymbol{(L)}}(\boldsymbol{\theta}) + N(\boldsymbol{0}\,, \boldsymbol{\Sigma}_{Kriging}(\boldsymbol{\theta})) \tag{C.8}$$

where $\boldsymbol{\Sigma_{Kriging}}$ is the covariance matrix associated with the Kriging prediction uncertainty, that is a $p^* \times p^*$ matrix having the mean square errors of each feature prediction as diagonal entries:

$$\boldsymbol{\Sigma}_{\boldsymbol{Kriging}} = \begin{bmatrix} \sigma^2_{z_1(\theta)} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma^2_{z_{p^*}(\theta)} \end{bmatrix} \tag{C.9}$$

5. finally, substitute equation (C.8) into (C.7) and assume that $N\left(\mathbf{0}\,,\boldsymbol{\Sigma}_{(L)}\right)$ and $N\left(\mathbf{0}\,,\boldsymbol{\Sigma}_{Kriging}\right)$ are statistically independent, then $p\left(\boldsymbol{z}^{\boldsymbol{E}}\middle|\boldsymbol{\theta}\right)$ can be written as:

$$p\left(\boldsymbol{z}^{\boldsymbol{E}}\middle|\boldsymbol{\theta}^*\right) = N\left(\,\hat{\boldsymbol{z}}^{\boldsymbol{MM}}_{(\boldsymbol{L})}(\boldsymbol{\theta}), \boldsymbol{\Sigma}_{(L)}(\boldsymbol{\theta}) + \boldsymbol{\Sigma}_{Kriging}(\boldsymbol{\theta})\right) \tag{C.10}$$

Since in this work $N_{exp} = 1$, considering equation (C.10), the posterior PDF reduces to:

$$p\left(\boldsymbol{\theta}\middle|\boldsymbol{z}^{\boldsymbol{E}}\right) \propto p(\boldsymbol{\theta}) \frac{1}{\left(\sqrt{2\pi}\right)^p \sqrt{|\boldsymbol{\Sigma}|}} exp\left[-\frac{1}{2}\left[\boldsymbol{z}^E - \hat{\boldsymbol{z}}^{\boldsymbol{MM}}_{(\boldsymbol{L})}(\boldsymbol{\theta})\right]^T \boldsymbol{\Sigma}^{-1} \left[\boldsymbol{z}^E - \hat{\boldsymbol{z}}^{\boldsymbol{MM}}_{(\boldsymbol{L})}(\boldsymbol{\theta})\right]\right] \tag{C.11}$$

where $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{(L)}(\boldsymbol{\theta}) + \boldsymbol{\Sigma}_{Kriging}(\boldsymbol{\theta})$. Note that $\boldsymbol{\Sigma}_{(L)}(\boldsymbol{\theta})$, in the expression (15) of Section 3.2, coincides with $\boldsymbol{\Sigma}_{exp}(\boldsymbol{\theta})$.

References

[1] D'Auria F, Bousbia-salah A, Petruzzi A, del Nevo A. State of the Art in Using Best Estimate Calculation Tools in Nuclear Technology. Nucl Eng Technol 2006;38:11–32.

[2] Iaea. Best Estimate Safety Analysis for Nuclear Power Plants: Uncertainty Evaluation; Safety Reports Series 52 2008:1–211.

[3] Ferson S, Ginzburg LR. Different methods are needed to propagate ignorance and variability. Reliab Eng Syst Saf 1996;54:133–44. https://doi.org/10.1016/S0951-8320(96)00071-3.

[4] Durga Rao K, Kushwaha HS, Verma AK, Srividya A. Quantification of epistemic and aleatory uncertainties in level-1 probabilistic safety assessment studies. Reliab Eng Syst Saf 2007;92:947–56. https://doi.org/10.1016/j.ress.2006.07.002.

[5] Winkler RL. Uncertainty in probabilistic risk assessment. Reliab Eng Syst Saf 1996;54:127–32. https://doi.org/10.1016/S0951-8320(96)00070-1.

[6] Apostolakis G. A commentary on model uncertainty 1994.

[7] Ferson S, Joslyn CA, Helton JC, Oberkampf WL, Sentz K. Summary from the epistemic uncertainty workshop: Consensus amid diversity. Reliab Eng Syst Saf 2004;85:355–69. https://doi.org/10.1016/j.ress.2004.03.023.

[8] Pourgol-Mohammad M. Thermal-hydraulics system codes uncertainty assessment: A review of the methodologies. Ann Nucl Energy 2009;36:1774–86. https://doi.org/10.1016/j.anucene.2009.08.018.

[9] Helton JC, Johnson JD. Quantification of margins and uncertainties: Alternative representations of

epistemic uncertainty. Reliab Eng Syst Saf 2011;96:1034–52. https://doi.org/10.1016/j.ress.2011.02.013.

[10] Shrestha R, Kozlowski T. Inverse uncertainty quantification of input model parameters for thermal-hydraulics simulations using expectation–maximization under Bayesian framework. J Appl Stat 2016;43:1011–26. https://doi.org/10.1080/02664763.2015.1089220.

[11] Katafygiotis LS, Beck JL. Updating Models and Their Uncertainties. II: Model Identifiability. J Eng Mech 1998;124:463–7. https://doi.org/10.1061/(asce)0733-9399(1998)124:4(463).

[12] Katafygiotis BLS, Beck JL. Updating models and their uncertainties. II: Model identifiability. J Eng Mech 1998;124:463–7.

[13] Gelman A, Carlin JB, Stern HS, Dunson DB, Vehtari A, Rubin DB. Bayesian data analysis 3rd ed. vol. 1542. 2015. https://doi.org/10.1017/CBO9781107415324.004.

[14] Kennedy MC, O'Hagan A. Bayesian calibration of computer models. J R Stat Soc Ser B (Statistical Methodol 2001;63:425–64. https://doi.org/10.1111/1467-9868.00294.

[15] Haftka T, Shyy W, Tucker PK. Surrogate-based Analysis and Optimization. 2020.

[16] Wang GG, Shan S. Review of metamodeling techniques in support of engineering design optimization. J Mech Des Trans ASME 2007;129:370–80. https://doi.org/10.1115/1.2429697.

[17] Wu X, Xie Z, Alsafadi F, Kozlowski T. A comprehensive survey of inverse uncertainty quantification of physical model parameters in nuclear system thermal–hydraulics codes. Nucl Eng Des 2021;384. https://doi.org/10.1016/j.nucengdes.2021.111460.

[18] Rasmussen CE, Williams CKI. Gaussian processes for machine learning. 2006. vol. 38. 2006.

[19] Lataniotis C, Wicaksono D, Marelli S, Sudret B. UQLab User Manual: Kriging (Gaussian Process Modeling) Report # UQLab-V1.3-105. Chair Risk, Saf Uncertain Quantif ETH Zurich, Switz 2019:1–18.

[20] Wilkinson RD. Bayesian Calibration of Expensive Multivariate Computer Experiments. Large-Scale Inverse Probl Quantif Uncertain 2010:195–215. https://doi.org/10.1002/9780470685853.ch10.

[21] Wang C, Wu X, Kozlowski T. Gaussian Process–Based Inverse Uncertainty Quantification for TRACE Physical Model Parameters Using Steady-State PSBT Benchmark. Nucl Sci Eng 2019;193:100–14. https://doi.org/10.1080/00295639.2018.1499279.

[22] Arendt PD, Apley DW, Chen W. Quantification of model uncertainty: Calibration, model discrepancy, and identifiability. J Mech Des Trans ASME 2012;134:1–12. https://doi.org/10.1115/1.4007390.

[23] Wu X, Kozlowski T, Meidani H, Shirvan K. Inverse uncertainty quantification using the modular Bayesian approach based on Gaussian Process, Part 2: Application to TRACE. Nucl Eng Des 2018;335:417–31. https://doi.org/10.1016/j.nucengdes.2018.06.003.

[24] Wu X, Kozlowski T, Meidani H, Shirvan K. Inverse uncertainty quantification using the modular Bayesian approach based on Gaussian process, Part 1: Theory. Nucl Eng Des 2018;335:339–55. https://doi.org/10.1016/j.nucengdes.2018.06.004.

[25] Fricker TE, Oakley JE, Urban NM. Multivariate gaussian process emulators with nonseparable covariance structures. Technometrics 2013;55:47–56. https://doi.org/10.1080/00401706.2012.715835.

[26] Kleijnen JPC, Mehdad E. Multivariate versus univariate Kriging metamodels for multi-response

simulation models. Eur J Oper Res 2014;236:573–82. https://doi.org/10.1016/j.ejor.2014.02.001.

[27] Conti S, O'Hagan A. Bayesian emulation of complex multi-output and dynamic computer models. J Stat Plan Inference 2010;140:640–51. https://doi.org/10.1016/j.jspi.2009.08.006.

[28] Mohammadi H, Challenor P, Goodfellow M. Emulating dynamic non-linear simulators using Gaussian processes. Comput Stat Data Anal 2019;139:178–96. https://doi.org/10.1016/j.csda.2019.05.006.

[29] Van Der Maaten LJP, Postma EO, Van Den Herik HJ. Dimensionality Reduction: A Comparative Review. J Mach Learn Res 2009;10:1–41. https://doi.org/10.1080/13506280444000102.

[30] Jolliffe IT. Principal Component Analysis. 2nd ed. Springer-Verlag New York; 2002.

[31] Higdon D, Gattiker J, Williams B, Rightley M. Computer model calibration using high-dimensional output. J Am Stat Assoc 2008;103:570–83. https://doi.org/10.1198/016214507000000888.

[32] Higdon D, Geelhood K, Williams B, Unal C. Calibration of tuning parameters in the FRAPCON model. Ann Nucl Energy 2013;52:95–102. https://doi.org/10.1016/j.anucene.2012.06.018.

[33] Wu X, Kozlowski T, Meidani H. Kriging-based inverse uncertainty quantification of nuclear fuel performance code BISON fission gas release model using time series measurement data. Reliab Eng Syst Saf 2018;169:422–36. https://doi.org/10.1016/j.ress.2017.09.029.

[34] Nagel JB, Rieckermann J, Sudret B. Principal component analysis and sparse polynomial chaos expansions for global sensitivity analysis and model calibration: Application to urban drainage simulation. Reliab Eng Syst Saf 2020;195:106737. https://doi.org/10.1016/j.ress.2019.106737.

[35] Roma G, Di Maio F, Bersano A, Pedroni N, Bertani C, Mascari F, et al. A Bayesian framework of inverse uncertainty quantification with principal component analysis and Kriging for the reliability analysis of passive safety systems. Nucl Eng Des 2021;379:111230. https://doi.org/10.1016/j.nucengdes.2021.111230.

[36] Zhao R, Yan R, Chen Z, Mao K, Wang P, Gao RX. Deep learning and its applications to machine health monitoring. Mech Syst Signal Process 2019;115:213–37. https://doi.org/10.1016/j.ymssp.2018.05.050.

[37] Holden AJ, Robbins DJ, Stewart WJ, Smith DR, Schultz S, Wegener M, et al. Reducing the Dimensionality of Data with Neural Networks 2006;313:504–7.

[38] Wang Y, Yao H, Zhao S. Auto-encoder based dimensionality reduction. Neurocomputing 2016;184:232–42. https://doi.org/10.1016/j.neucom.2015.08.104.

[39] Monisha R, Mrinalini R, Britto MN, Ramakrishnan R, Rajinikanth V. Smart Intelligent Computing and Applications. vol. 104. 2019. https://doi.org/10.1007/978-981-13-1921-1.

[40] Mao X-J, Shen C, Yang Y-B. Image Restoration Using Convolutional Auto-encoders with Symmetric Skip Connections 2016:1–17.

[41] Olshausen BA, Fieldt DJ. Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1 ? Coding V1 Gabor-wavelet Natural images. Vis Res 1997;37:3311–25.

[42] Vincent P, Larochelle H. Extracting and Composing Robust Features with Denoising.pdf 2008:1096–103.

[43] Kingma DP, Welling M. Auto-encoding variational bayes. 2nd Int Conf Learn Represent ICLR 2014 - Conf Track Proc 2014:1–14.

[44] Ng A. Sparse autoencoder, CS294A Lecture notes, 2011, p. 1–19.

[45] Utgoff PE, Stracuzzi DJ. Many-layered learning. Proc - 2nd Int Conf Dev Learn ICDL 2002 2002:141–6. https://doi.org/10.1109/DEVLRN.2002.1011824.

[46] Yang Z, Baraldi P, Zio E. Automatic Extraction of a Health Indicator from Vibrational Data by Sparse Autoencoders. Proc - 2018 3rd Int Conf Syst Reliab Safety, ICSRS 2018 2019:328–32. https://doi.org/10.1109/ICSRS.2018.8688720.

[47] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature 1986;323:533–6. https://doi.org/10.1038/323533a0.

[48] McKay MD, Beckman RJ, Conover WJ. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 2000;42:55–61. https://doi.org/10.1080/00401706.2000.10485979.

[49] Welch G, Bishop G. An Introduction to the Kalman Filter. In Pract 2006;7:1–16. https://doi.org/10.1.1.117.6808.

[50] Idaho National Laboratory. RELAP5-3D Code Manual Volume I: Code Structure, System Models and Solution Methods, 2015 2015.

[51] Bersano A, Bertani C, Falcone N, de Salve M, Mascari F, Meloni P. Qualification of RELAP5-3D code against the in-pool passive energy removal system PERSEO data. 30th Eur Saf Reliab Conf ESREL 2020 15th Probabilistic Saf Assess Manag Conf PSAM 2020 2020:1150–7.

[52] Mascari F, Lombardo C, De Salve M, Bertani C, Bersano A, Falcone N, et al. Description of PERSEO Test n. 7 for International Open Benchmark Excercise, ADPFISS-LP1-126 2019.

[53] Ferri R, Achilli A, Cattadori G, Bianchi F, Meloni P. Design, experiments and Relap5 code calculations for the perseo facility. Nucl Eng Des 2005;235:1201–14. https://doi.org/10.1016/j.nucengdes.2005.02.011.

[54] Bandini G, Meloni P, Polidori M, Lombardo C. Validation of CATHARE V2.5 thermal-hydraulic code against full-scale PERSEO tests for decay heat removal in LWRs. Nucl Eng Des 2011;241:4662–71. https://doi.org/10.1016/j.nucengdes.2011.02.034.

[55] Saltelli A, Annoni P, Azzini I, Campolongo F, Ratto M, Tarantola S. Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index. Comput Phys Commun 2010;181:259–70. https://doi.org/10.1016/j.cpc.2009.09.018.

[56] Glaeser H. GRS method for uncertainty and sensitivity evaluation of code results and applications. Sci Technol Nucl Install 2008;2008. https://doi.org/10.1155/2008/798901.

[57] Wilks S. Determination of sample sizes for setting tolerance limits. Ann Math Stat 1941;12:91–6.

[58] Guba A, Makai M, Pál L. Statistical aspects of best estimate method - I. Reliab Eng Syst Saf 2003;80:217–32. https://doi.org/10.1016/S0951-8320(03)00022-X.

[59] Dubourg V. Méta-modèles adaptatifs pour l’analyse de fiabilité et l’optimisation sous contrainte fiabiliste 2011:308.

[60] Helton JC, Davis FJ. Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. Reliab Eng Syst Saf 2003;81:23–69. https://doi.org/10.1016/S0951-8320(03)00058-9.

[61] Bachoc F. Cross validation and maximum likelihood estimations of hyper-parameters of Gaussian processes with model misspecification. Comput Stat Data Anal 2013;66:55–69. https://doi.org/10.1016/j.csda.2013.03.016.

[62] Martin JD, Simpson TW. Use of kriging models to approximate deterministic computer models. AIAA J 2005;43:853–63. https://doi.org/10.2514/1.8650.

[63] Wall ME, Rechtsteiner A, Rocha LM. Singular Value Decomposition and Principal Component Analysis. A Pract Approach to Microarray Data Anal 2005:91–109. https://doi.org/10.1007/0-306-47815-3_5.

[64] Haario H, Saksman E, Tamminen J. An adaptive Metropolis algorithm. Bernoulli 2001;7:223–42. https://doi.org/10.2307/3318737.

[65] Wagner P-R, Nagel J, Marelli S, Sudret B. UQLab user manual: Bayesian inference for model calibration and validation 2019.

[66] Cox DR, Hinkley D V, Rubin D, Silverman BW, Bartlett MS, Beard RE, et al. Density Estimation for Statistics and Data Analysis. 1986.

[67] Abdelaziz AH, Watanabe S, Hershey JR, Kolossa D, Abdelaziz AH, Watanabe S, et al. Uncertainty propagation through deep neural networks To cite this version : Others 2015.

[68] Hadjahmadi AH, Homayounpour MM. Uncertainty propagation through neural network bottleneck features. ICEE 2015 - Proc 23rd Iran Conf Electr Eng 2015;10:567–75. https://doi.org/10.1109/IranianCEE.2015.7146280.

[69] Titensky JS, Jananthan H, Kepner J. Uncertainty Propagation in Deep Neural Networks Using Extended Kalman Filtering. 2018 IEEE MIT Undergrad Res Technol Conf URTC 2018 2018. https://doi.org/10.1109/URTC45901.2018.9244804.