Efficient Global Sensitivity Analysis of Model-Based Ultrasonic Nondestructive Testing Systems using Machine Learning and Sobol' Indices (Invited)

Jethro Nagawkar
Ph.D. Student, Member of ASME
Department of Aerospace Engineering
Iowa State University
Ames, Iowa, 50011
Email: jethro@iastate.edu

Leifur Leifsson*

Associate Professor, Member of ASME
Department of Aerospace Engineering
Iowa State University
Ames, Iowa, 50011
Email: leifur@iastate.edu

The objective of this work is to reduce the cost of performing model-based sensitivity analysis for ultrasonic nondestructive testing systems by replacing the accurate physics-based model with machine learning (ML) algorithms and quickly compute Sobol' indices. The ML algorithms considered in this work are neural networks (NN), convolutional NN (CNN), and deep Gaussian processes (DGP). The performance of these algorithms is measured by the root mean squared error on a fixed number of testing points and by the number of high-fidelity samples required to reach a target accuracy. The algorithms are compared on three ultrasonic testing benchmark cases with three uncertainty parameters, namely, spherically-void defect under a focused and a planar transducer and spherical-inclusion defect under a focused transducer. The results show that NN required 35, 100, and 35 samples for the three cases, respectively. CNN required 35, 100, and 56, respectively, while DGP required 84, 84, and 56, respectively.

Keywords: ultrasonic nondestructive testing, Sobol' indices, machine learning, neural networks, convolutional neural networks, deep Gaussian processes

1 Introduction

Nondestructive testing (NDT) [1, 2] is the process of evaluating, testing or inspecting assemblies or components for discontinuities or damages without affecting the serviceability of the part. NDT can be performed either during manufacturing or while the component is in service. This is essential to ensure product integrity and reliability as well as lower production cost and maintaining a uniform level of product quality. NDT has been successfully used in various applications such as aircraft damage estimation [3, 4], weld defect inspection [5, 6], and flaw characterization [7, 8].

NDT measurements have traditionally relied on experimental methods. These methods, however, can be both time-consuming as well as costly to perform. In order to reduce this time and cost, various physics-based NDT models [9–11] have been developed and used to reduce the need for empirical data. These include numerical methods such as finite element methods [12, 13] and boundary element methods [14, 15], as well as approximation methods such as Gaussian beam superposition methods [16, 17] and ray tracing methods [18, 19]. The aforementioned physics-based models can be cheaper, both financially and computationally, when compared to experimental methods, and can also have high accuracy. This can prove essential to reducing the need for empirical data, by replacing experimental methods with physics-based models.

Sensitivity analysis [20, 21] is an approach to quantify the effect of each individual input parameter or combinations of input parameters on the system response. It can be classified as either a local [22, 23] or a global sensitivity analysis [24, 25]. In local sensitivity analysis, small perturbations

*Address all correspondence to this author.

in the input parameter space are used to quantify their effects on the system response. For global sensitivity analysis, the variance of the system response due to the input parameter variability is quantified.

In this study, global variance-based sensitivity analysis with Sobol' indices [26, 27] is used to quantify the effects of input parameter variability of the output of physics-based ultrasonic testing (UT) simulations. The main elements of the model-based sensitivity analysis are (1) identifying the important input variability parameters and their corresponding probability distributions, (2) propagating these input parameters through the physics-based model, called uncertainty propagation, and (3) performing sensitivity analysis using Sobol' indices.

The key challenges of model-based sensitivity analysis include (1) each physics-based model can be computationally expensive to solve, (2) a large number of variability parameters can exist for the NDT systems, and (3) multiple and repetitive physics-based model evaluations are required for sensitivity analysis. This results in problems that are challenging to solve in a reasonable amount of time.

To alleviate this computational cost, various metamodeling methods [28, 29] have been developed. Metamodeling methods replace the time-consuming and accurate physics-based models with a computationally efficient metamodel (also called surrogate model). Metamodeling methods can be broadly classified as either data-fit methods [28,30] or multi-fidelity methods [31,32]. Data-fit methods are constructed by fitting a response surface through evaluated model responses at sampled high-fidelity data points. Examples of data-fit methods include polynomial chaos expansions (PCE) [33], Kriging [34], and support vector machines [35]. In multifidelity methods, on the other hand, low-fidelity data is used to enhance the prediction capabilities of a data-fit model constructed from a limited number of high-fidelity data points. Examples of such methods include Cokriging [36] and manifold mapping [37].

Metamodeling methods have been utilized for various NDT applications. Bilicz et al. [38, 39] used Kriging [34] for both forward and inverse problems using eddy current NDT. Support vector regression [35] was used by Miorelli et al. [40] to perform both sensitivity analysis and probability of detection for eddy current NDT systems. Du et al. [41] performed sensitivity analysis and probability of detection for UT NDT systems using PCE [42] and Kriging [34]. Du and Leifsson [43] developed the polynomial chaos-based Cokriging multifidelity method [43] and used it for model-based probability of detection in UT NDT systems.

This paper introduces and applies three different machine learning algorithms for sensitivity analysis of UT NDT systems. In particular, neural networks (NN) [44, 45], convolutional neural networks (CNN) [46, 47], and deep Gaussian processes (DGP) [48, 49] are introduced to the global variance-based sensitivity analysis of UT NDT systems. Note that for these applications, the machine learning algorithms used can be classified as data-fit metamodeling methods. The UT NDT cases considered in this study are three benchmark cases developed by the World Federation of Nondestructive Evaluation Centers (WFNDEC)[1]. The benchmark cases are (1) spherically-void defect under a focused transducer, (2) spherically-void defect under a planar transducer, and (3) spherical-inclusion defect under a focused transducer. The sensitivity analysis results from the machine learning algorithms are compared to those obtained from directly evaluating the high-fidelity physics-based model. In this work, the analytical UT model by Thompson and Gray [50, 51] is used as the high-fidelity physics-based model.

The remainder of this paper is organized as follows. The following section describes the methods used in this paper to train the machine learning algorithms and perform sensitivity analysis. The next section presents the result of the application of these algorithms to the benchmark cases. The paper ends with the conclusion of this study and provides suggestions for future work.

2 Methods

The methods used in this work are described in this section. The workflow of the model-based sensitivity analysis is described first, followed by the sampling plan, a detailed description of the machine learning algorithms, their validation and finally, the sensitivity analysis with Sobol' indices.

2.1 Workflow

A flowchart of the model-based sensitivity analysis is shown in Fig. 1. The process begins by sampling the input parameter space. Two separate sets of sampling plans are created, one for training and another for testing. The high-fidelity physics-based model simulations are then evaluated using those sampling plans to gather the output responses. The training data is then used to train the machine learning algorithms and the accuracy of these algorithms are tested using the testing data. If this accuracy, measured in terms of the root mean squared error (RMSE), does not meet the threshold of 1% standard deviation of the testing points (σ_t), a new training set with a higher number of sample points is created and the previous steps are repeated. Once the machine learning-based models are accurate enough, sensitivity analysis with Sobol' indices [26] is performed.

2.2 Sampling plan

The process of selecting discrete samples in the input variability space is known as sampling. It is an iteration-based process in which the input variability parameters are randomly drawn from probability distributions assigned to the parameters. In this work, Latin Hypercube sampling (LHS) [52] is used to generate the training plan, while Monte Carlo sampling (MCS) [53] is used to generate the testing plan. MCS is also used to generate the sampling plan for the Sobol' indices [26] calculation.

MCS [53] starts by generating random numbers within the [0,1] interval with replacement. These random numbers

[1] https://www.wfndec.org/benchmark-problems/

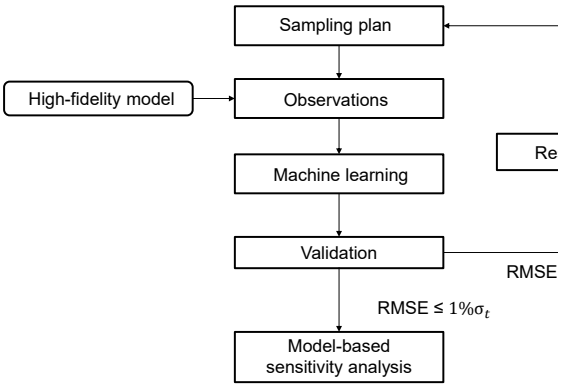

Fig. 1. A flowchart of the machine learning-based sensitivity analysis

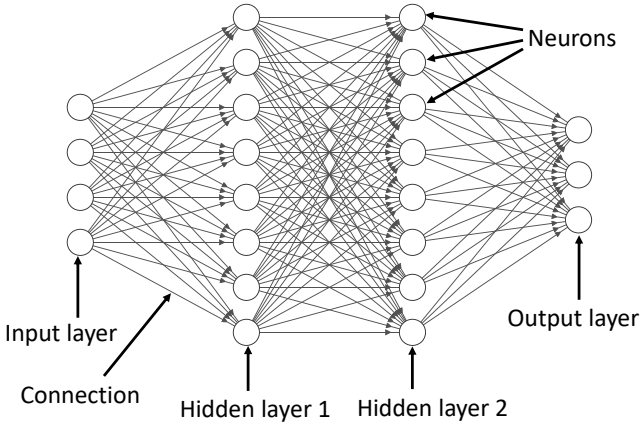

Fig. 2. A schematic of a neural network

are used as probabilities of the associated cumulative density functions of the variability parameters. The corresponding values can then be obtained using quantile functions. LHS [52, 54] on the other hand aims at sampling the variability parameters more effectively than MCS. This is done by stratifying the probability distributions into equal intervals in the [0,1] range. Random samples are then selected from each interval. This prevents the clustering of the generated numbers. The remaining steps are the same as MCS.

2.3 Neural networks

NN [46] methods can be classified as a subclass of data-fit metamodeling methods. Any complex function can be approximated through an hierarchy of features. NN [45] have multiple steps in the hierarchy, which starts with an input and ends with an output. Each step is known as a layer. The layers in-between the inputs and the outputs are called hidden layers [44]. Figure 2 shows an example of a schematic of a NN architecture with two hidden layers. The input and output layer size depend on the dimensionality of the input and output for a given problem. For the three UT benchmark cases, the input has three features, while the output has a size of one. Each hidden layer in a NN consists of neurons, which are a fundamental unit of computation in a NN [46]. Neurons calculate a weighted sum of the outputs from a previous hidden or input layer and outputs a nonlinear transformation of it. This nonlinear transformation is termed activation. In Fig. 2, each hidden layer has eight neurons. By changing the number of hidden layers as well as the number of neurons in each hidden layer, the NN can approximate functions of arbitrary complexity [46].

The activation function in each neuron of a given hidden layer, L, is given by [46]

$$z_j^{(L)} = a\Big(\sum_{i=1}^{N^{(L-1)}} \omega_{ji}^{(L)} z_i^{(L-1)} + b^{(L-1)}\Big), \tag{1}$$

where a is the activation function, $\omega_{ji}^{(L)}$ is the weight between the i^{th} neuron in the $L-1$ hidden layer and j^{th} neuron in the L hidden layer, and $b^{(L-1)}$ is the bias unit in the $L-1$ hidden layer. The weight and biases together are termed the parameters of the NN and are tuned using a gradient-based optimizer [46]. Here, the maximum value of i is $N^{(L-1)}$, that is the number of neurons in the $L-1$ hidden layer. $z_j^{(L)}$ is the output of the j^{th} neuron in the L hidden layer, and $z_i^{(L-1)}$ is the output of the i^{th} neuron in the $L-1$ hidden layer.

A loss function, $\mathcal{L}$, is defined to capture the mismatch between the training data observations, given by y, and the predicted value of the NN, given by $\hat{y}$ [46]. The loss function chosen in this study is the mean squared error between y and $\hat{y}$ and is averaged over all the training data. The loss function is written as

$$\mathcal{L} = \frac{\sum_{l=1}^{N_{tr}} \sum_{m=1}^{N_o} (\hat{y}_m^{(l)} - y_m^{(l)})^2}{N_{tr} N_o}, \tag{2}$$

where N_o is the size of the output layer and N_{tr} is the number of training data sets, and m is the index of the neuron in the output layer. In practice, averaging of the loss function is generally performed over a subset of the training data sets, known as mini-batch [46].

Training the NN involves solving an optimization problem where the objective function is the loss function and is minimized to improve the prediction capabilities of the NN. The gradient of this objective function is calculated with respect to the parameters of the NN and is done efficiently using the backpropagation algorithm [55]. The optimizer used is Adaptive Moments (ADAM) algorithm [56] and has the following steps:

1. Update the biased first moment estimate:

$$\mathbf{V}_k \leftarrow \beta_1 \mathbf{V}_{k-1} + (1-\beta_1)\mathbf{G}_k, \tag{3}$$

where $\mathbf{G}_k$ is the gradient of the loss function with respect to the parameters at a given iteration k, $\mathbf{V}_k$ is the exponential moving averages of the gradients (also called biased first moment estimate), and β_1 is the exponential

decay rate for $\mathbf{V}_k$. The recommended value for β_1 is 0.9 [56] and is used in this study.

2. Update the biased second moment estimate:

$$\mathbf{S}_k \leftarrow \beta_2 \mathbf{S}_{k-1} + (1-\beta_2)\mathbf{G}_k^2, \tag{4}$$

where $\mathbf{S}_k$ is the exponential moving averages of the squared gradients (also called biased second moment estimate), and β_2 is the exponential decay rate for $\mathbf{S}_k$. 0.999 is the recommended value for β_2 [56] and is used this study.

3. Correct the bias in the first moment:

$$\widehat{\mathbf{V}}_k \leftarrow \frac{\mathbf{V}_k}{1-\beta_1^k}, \tag{5}$$

where the bias introduced by setting $\mathbf{V}_0$ to zero is corrected by $\widehat{\mathbf{V}}_k$.

4. Correct the bias in the second moment:

$$\widehat{\mathbf{S}}_k \leftarrow \frac{\mathbf{S}_k}{1-\beta_2^k}, \tag{6}$$

where the bias introduced by setting $\mathbf{S}_0$ to zero is corrected by $\widehat{\mathbf{S}}_k$.

5. Update the parameters:

$$\theta_{k+1} \leftarrow \theta_k + \alpha_k \frac{\widehat{\mathbf{V}}_k}{\sqrt{\widehat{\mathbf{S}}_k} + \eta}, \tag{7}$$

where α_k is the learning rate, θ_k are the NN parameter values, and η is a small value that is specified in order to prevent the denominator from being zero.

The hyperparameters of a NN include the number of hidden layers, number of neurons in each hidden layer, the mini-batch size, the maximum number of epochs, the learning rate and the activation function. Rigorous rules for selecting the hyperparameter settings for an NN do not exist. In this work, various hyperparameter settings were selected by iterating over them. The hyperparameters chosen were the ones that resulted in the lowest RMSE as described in Section 2.6. The NN architecture used in this work includes one hidden layer with 50 neurons. The mini-batch size selected is 10. Maximum number of epochs is set to 10,000. One epoch refers to one iteration over an entire training data set [44]. The learning rate selected is 0.01. The activation function chosen for this study is the hyperbolic tangent function [46]. To construct the NN, the Keras [57] wrapper with Tensorflow [58] is used in this study.

2.4 Convolutional neural networks

CNN [46] are a special type of NN that are used to process data with a grid-like topology [46], such as images. In images, each grid location contains pixel values, and when combined together results in the final image. For this study, the variability parameters are used in the place of the pixels. Since CNN employs the mathematical operation called convolution, it is named as such [46]. Any NN with at least one convolutional layer is termed CNN. For image recognition tasks, the number of parameters in a NN can grow really fast, as it depends on the number of input features. For images, the size of the image defines the number of input features. In CNN, the number of parameters is independent of the number of input features and is dependant on the size and number of filters of a convolutional kernel [46]. This reduces the number of parameters to be tuned and prevents overfitting in the presence of limited data.

Figure 3 shows a schematic of a CNN architecture which contains one convolutional layer, followed by one max-pooling layer, two fully connected layers and an output layer. The output of a convolutional layer or max-pooling layer is termed feature maps. Feature maps are similar to images and also have a grid-like topology with pixel values. The number of channels, also known as depth of a feature map, depends on the hyperparameter called number of filters. In Fig. 3, this value is four. To convert a feature map to a fully connect layer, the flatten operation is performed. Flatten converts the three dimensional feature map into a one dimensional fully connected layer. Similar to a NN, the number of these layers can vary.

The input image in Fig. 3 has one channel (grey image) and has 32 pixels (grids) in the horizontal and vertical directions each. Note that colored images have three channels, namely, red, blue and green, respectively. The kernel or filter, which contains parameters to be tuned, has five grids in the horizontal and vertical directions each, that is a 5×5 kernel. In a convolutional operation, an element-wise product between the kernel parameter value and image pixel value is performed. This process continues by moving the kernel over the image, from left to right, top to bottom. The hyperparameter stride, is used to define how many pixels in both the horizontal and vertical directions to move after performing one convolutional operation.

The output of a convolutional operation is given by

$$c_{i,j}^{(L_c)} = a\Big(\sum_{m=1}^{N_{c,v}} \sum_{n=1}^{N_{c,h}} f_{m,n} c_{i+m-1,j+n-1}^{(L_c-1)} + b^{(L_c-1)}\Big), \tag{8}$$

where a is the activation function, $N_{c,h}$ and $N_{c,v}$ refer to the number of grids in the convolutional kernel in the horizontal and vertical directions, respectively, $f_{m,n}$ is the weight of the kernel at the m^{th} row and n^{th} column of the grid in the kernel, $c^{(L_c)}$ is the output of the convolution in the L_c convolutional layer, and $c^{(L_c-1)}$ is the input into the L_c convolutional layer. This value could be from either a max-pooling layer or a convolutional layer. The output of a max-pooling layer is given by

$$p_{i,j}^{(L_p)} = \max_{1 \le m \le N_{p,v}, 1 \le n \le N_{p,h}} \Big(p_{i+m-1,j+n-1}^{(L_p-1)}\Big), \tag{9}$$

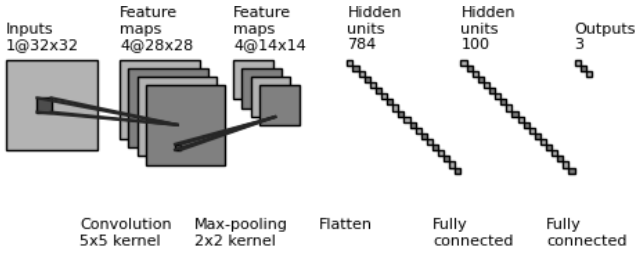

Fig. 3. A schematic of a convolutional neural network

where $N_{p,h}$ and $N_{p,v}$ refer to the number of grids in the max-pooling kernel in the horizontal and vertical directions, respectively, $p^{(L_p)}$ is the output of the L_p max-pooling layer, and $p^{(L_p-1)}$ is the input into the L_p max-pooling layer. The max-pooling kernel selects the maximum pixel value and unlike the convolutional kernel has no parameters. Max-pooling is performed to reduce the number of parameter in a CNN [46].

Training the CNN is similar to training the NN. The loss function described in (2) is used as the objective function for training the CNN. The gradients are calculated using the backpropagation algorithm [55] and the ADAM [56] optimizer is used to minimize the loss function.

The hyperparameters used in the CNN are the number of convolutional and max-pooling layers, the convolutional and max-pooling kernel size. The number of filters of each convolutional kernel and the stride of each kernel is considered as well. Other hyperparameters include the number of fully connected layers, the number of neurons in each layer, the mini-batch size, the maximum number of epochs, the learning rate and the activation function. The CNN used in this work includes one convolutional layer, one fully connected layer and no max-pooling layer. Only one convolutional kernel of size 1×1 is selected and has a stride of value one. The mini-batch selected has a size of 10, while the maximum number of epochs is set to 10,000. The number of neurons in the fully connected layer is 100 and the learning rate of the ADAM [56] optimizer is 0.01. Hyperbolic tangent function [46] is selected as the activation function for this study. The process of selecting these hyperparameters is similar to those used for the NN. To construct the CNN, the Keras [57] wrapper with Tensorflow [58] is used in this study.

One final thing to note is that, while images are not used, each variability parameter is assigned to each grid and the convolutional operation is performed on it. The input grid therefore has a size of 3×1. While this grid size is lower compared to most image sizes, CNN can be easily expanded to work on NDT cases with higher number of variability parameters.

2.5 Deep Gaussian processes

DGP [48] are multi-layer generalizations of Gaussian processes (GP), where the inputs to one GP is from the outputs of another. The architecture is similar to a NN (Fig. 2), however, the activation functions in the neurons are replaced by GP mappings. DGP are shown to overcome the limitations of single-layer GP, while retaining its advantages [49]. DGP are shown to work well on limited amount of data [48], which is useful for NDT sensitivity analysis.

Consider a GP mapping given by

$$z_i = f_i(\mathbf{X}) + \varepsilon_i, \tag{10}$$

where $\mathbf{X} \in \mathbb{R}^m$ is the vector of m-dimensional input variability parameters, f is a zero mean GP mapping: $f \sim GP(0, k(\mathbf{X}, \mathbf{X}))$, ε is the identically and independently distributed Gaussian noise ($N(0, \sigma_\varepsilon^2)$), and z_i is the output of the i^{th} neuron in the hidden layer. The covariance function, k, uses a Gaussian correlation [59] function (also known as automatic relevance determination correlation function [48]) and is given by

$$k(\mathbf{X}, \mathbf{X}') = \sigma^2 \exp\left[-\sum_{k=1}^{l}\left(\frac{X_k - X_k'}{h_k}\right)^2\right], \tag{11}$$

where σ and h_k are hyperparameters that need to be tuned, and l is the number of inputs to the neuron in the hidden layer. Another covariance function, the Matern-5/2 [60] correlation function, given by

$$\begin{aligned} &k(\mathbf{X}, \mathbf{X}') = \\ &\sigma^2\left[1 + \sqrt{5}\sqrt{\sum_{k=1}^{l}\left(\frac{X_k - X_k'}{h_k}\right)^2} + \frac{5}{3}\sum_{k=1}^{l}\left(\frac{X_k - X_k'}{h_k}\right)^2\right] \\ &\exp\left[-\sqrt{5}\sqrt{\sum_{k=1}^{l}\left(\frac{X_k - X_k'}{h_k}\right)^2}\right], \end{aligned} \tag{12}$$

is used in this work.

Damianou and Lawrence [48] developed a Bayesian training framework[2] to train all the parameters of the DGP. This framework is used to construct the DGP in the present work. The details of this framework are beyond the scope of this paper and can be found in [48].

The hyperparameters used in the DGP include the number of hidden layers, the number of GP mappings in each hidden layer, the correlation used as the GP mapping, and the total number of training iterations. Similar to NN and CNN, these hyperparameters where chosen that resulted in the lowest RMSE. For this study, the DGP used had one hidden layer, with a Matern-5/2 correlation [60] function for the hidden and output layer. The total number of iterations used was set to 1,500.

2.6 Validation

The global accuracy of the machine learning algorithms used in this work is measured using the RMSE, given by

$$\text{RMSE} = \sqrt{\textstyle\sum_{i=1}^{N_t}(\hat{y}_t^{(i)} - y_t^{(i)})^2 / N_t}, \tag{13}$$

[2] https://github.com/SheffieldML/PyDeepGP

and the normalized RMSE (NRMSE), given by

$$\text{NRMSE} = \text{RMSE}/(\max(\mathbf{y}_t) - \min(\mathbf{y}_t)), \tag{14}$$

where N_t is the total number of testing data, and $\hat{y}_t^{(i)}$ and $y_t^{(i)}$ are the machine learning estimation and high-fidelity observation of the i^{th} testing point, respectively. The maximum and minimum values of the high-fidelity physics-based model observations of the testing points are $\max(\mathbf{y}_t)$ and $\min(\mathbf{y}_t)$, respectively. An RMSE less than or equal to $1\%\sigma_t$ is considered an acceptable global accuracy criterion in this work.

2.7 Model-based sensitivity analysis

Sensitivity analysis with Sobol' indices [27] is used to quantify the effect of each input variability parameter, as well as a combination of input parameters, on the output model response. A black-box model

$$M(\mathbf{X}) = f(\mathbf{X}), \tag{15}$$

where $\mathbf{X}$ is the m variable random input vector, can be decomposed as [26]

$$\begin{aligned} M(\mathbf{X}) = & f_0 + \sum_{i=1}^{m} f_i(X_i) + \sum_{i<j}^{m} f_{i,j}(X_i, X_j) \\ & + \ldots + f_{1,2,\ldots,m}(X_1, X_2, \ldots, X_m), \end{aligned} \tag{16}$$

where f_0 is a constant, and f_i is a function of X_i and so on. One condition of this functional decomposition is that all the terms need to be orthogonal, which can then be decomposed in terms of the conditional expected values

$$f_0 = \mathbb{E}(M(\mathbf{X})), \tag{17}$$

$$f_i(X_i) = \mathbb{E}(M(\mathbf{X})|X_i) - f_0, \tag{18}$$

and

$$\begin{aligned} f_{i,j}(X_i, X_j) = & \mathbb{E}(M|X_i, X_j) - f_0 \\ & - f_i(X_i) - f_j(X_j), \end{aligned} \tag{19}$$

and so on. Here, f_i refers to the effect of varying individual input parameter X_i alone. This is termed the main effect of X_i. $f_{i,j}$ is the effect of varying X_i and X_j simultaneously and is called the second-order interaction. Higher-order interactions have analogous definitions. The variance of (16) is then

$$\mathbb{V}ar(M(\mathbf{X})) = \sum_{i=1}^{m} V_i + \sum_{i<j}^{m} V_{i,j} + \ldots + V_{1,2,\ldots,m}, \tag{20}$$

where

$$V_i = \mathbb{V}ar_{X_i}(\mathbb{E}_{\mathbf{X}_{\sim i}}(M(\mathbf{X})|X_i)), \tag{21}$$

$$V_{i,j} = \mathbb{V}ar_{X_{i,j}}(\mathbb{E}_{\mathbf{X}_{\sim i,j}}(M(\mathbf{X})|X_i, X_j)) - V_i - V_j, \tag{22}$$

and so on. The $\mathbf{X}_{\sim i}$ notation denotes the set of all variables except X_i. V_i refers to the variance of the output due to X_i alone, while $V_{i,j}$ is the variance of the output due to second-order interactions.

The main effect indices, given by the first-order Sobol' indices [26], are

$$S_i = \frac{V_i}{\mathbb{V}ar(M(\mathbf{X}))}, \tag{23}$$

where S_i measures the contribution of each individual X_i on the output variance. The total-order indices, given by the total-effect Sobol' indices [26], are

$$\begin{aligned} S_{T_i} &= \frac{\mathbb{E}_{\mathbf{X}_{\sim i}}(\mathbb{V}ar_{X_i}(M(\mathbf{X})|\mathbf{X}_{\sim i}))}{\mathbb{V}ar(M(\mathbf{X}))} \\ &= 1 - \frac{\mathbb{V}ar_{\mathbf{X}_{\sim i}}(\mathbb{E}_{X_i}(M(\mathbf{X})|\mathbf{X}_{\sim i}))}{\mathbb{V}ar(M(\mathbf{X}))}, \end{aligned} \tag{24}$$

where S_{T_i} is the measure of the output variance due to X_i alone as well as due to the interaction of X_i with other input parameters.

3 Numerical examples

In this section, the three machine learning algorithms are applied to three UT benchmark cases developed by the WFNDEC. The accuracy of the sensitivity analysis results are compared to those obtained from directly sampling the high-fidelity physics-based models.

3.1 Description of the benchmark cases

The three UT benchmark cases considered are the spherically-void defect under a focused transducer (Case 1), spherically-void defect under a planar transducer (Case 2), and the spherical-inclusion defect under a focused transducer (Case 3). The setup for these cases are shown in Fig. 4. Each of these cases have three variability parameters as inputs. Cases 1 and 3 have the probe angle (θ), the x-coordinates (x) and the frequency related F-number (F) as variability parameters. For Case 2, the F-number is replaced with the y-coordinates (y). Table 1 lists the variability parameters along with their input distributions. The output response is the voltage waveform at the receiver.

The Thompson-Grey analytical model [50] is used as the high-fidelity physics-based model in this study. The center frequency of the transducer is 5 MHz. The density of the

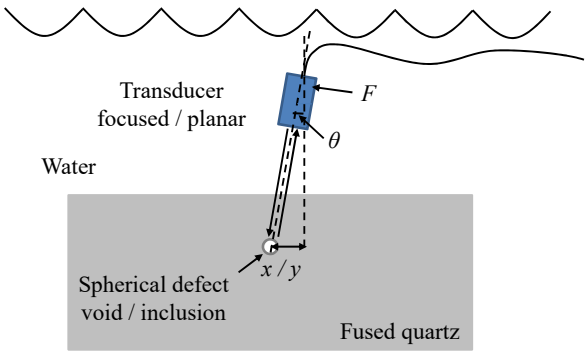

Fig. 4. Setup of the ultrasonic testing system for the benchmark cases

Table 1. The variability parameters used in the numerical examples

Parameters	Case 1	Case 2	Case 3
θ (deg)	$N(0, 0.5^2)$	$N(0, 0.5^2)$	$N(0, 0.5^2)$
x (mm)	$U(0, 1)$	$U(0, 1)$	$U(0, 1)$
y (mm)	N/A	$U(0, 1)$	N/A
F	$U(13, 15)$	N/A	$U(8, 10)$

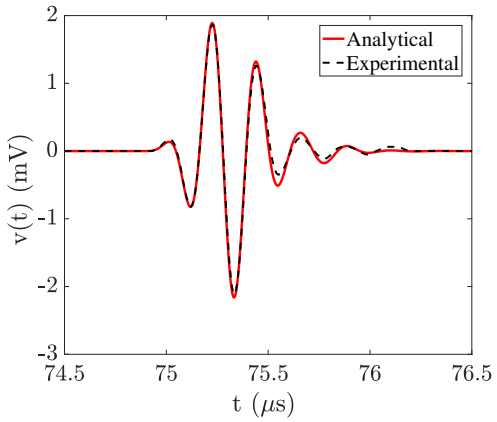

Fig. 5. Spherically-void defect under a planar transducer output response model validation [41]

fused quartz block is $2,000\ kg/m^3$. The longitudinal wave speed is $5,969.4\ m/s$, while the shear wave speed is $3,774.1\ m/s$. A detailed description of the UT testing model can be found in Schmerr and Song [51], while the validation of this model with experimental data can be found in Du et al. [41]. Figure 5 shows the validation of the time-domain waveform obtained from the high-fidelity physics-based model and is compared to experimental data for Case 2. The physics-based model matches the experimental results well.

3.2 Results

In Section 2, as discussed, the machine learning algorithms are required to have a global accuracy, measured in terms of RMSE, of less than $1\%\sigma_t$, before performing sensitivity analysis. For the three UT cases, the convergence of the RMSE with increasing number of high-fidelity training points is performed at a single defect radius size (a) of 0.5 mm and is shown in Figs. 6 to 8. For each machine learning algorithm and for each high-fidelity sample size, ten different LHS are generated to account for the variation in the input variability space. The RMSE plots in Figs. 6 to 8 show both the mean as well as the standard deviation in the RMSE arising due to the different input samples generated. In all these cases, the number of testing points is fixed and contains 1,000 high-fidelity MCS. Figures 6 to 8 also show the $10\%\sigma_t$ and $1\%\sigma_t$ values.

In Figs. 6 to 8, the RMSE decreases with increasing number of high-fidelity sample points. For Case 1, both NN and CNN perform similarly and require around 35 high-fidelity sample points to reach the target global accuracy. DGP for this case requires around 84 high-fidelity samples to reach this same threshold. For Case 2, DGP requires 84 high-fidelity samples and outperforms both NN and CNN, which require 100 high-fidelity samples each. Finally, in Case 3, NN outperforms both CNN and DGP. NN requires 35 high-fidelity samples, while the other two machine learning algorithms require 56 samples each. Using the same number of high-fidelity samples as those required to reach the global accuracy, the machine learning algorithms were trained for different defect sizes ranging from 0.1 mm to 0.5 mm and this error is now measured using NRMSE.

Figures 9 to 11 show the NRMSE as a variation in the defect size for the three UT cases. For all these cases, NRMSE is nearly constant and within the $1\%\sigma_t$ global accuracy, indicating that these machine learning algorithms are robust under varying defect sizes. Note that similar to the RMSE convergence plots, the NRMSE plots are generated using ten different samples at each defect size and for each machine learning algorithm. Figures 9 to 11 show both the mean and the standard deviation in the NRMSE.

To perform the sensitivity analysis with Sobol' indices, ten different sets containing 75,000 MCS each were generated for each of the machine learning algorithms and for each UT case. This analysis was only performed at a defect size of $a = 0.5$ mm. The trained machine learning algorithms were used to provide the model response for all the generated MCS. These model responses were then used to calculate the Sobol' indices. The same number of MCS were also generated to directly evaluate the physics-based models in order to perform sensitivity analysis. Figures 12 to 14 shows the 1^{st}-order Sobol' indices for the three UT cases, while Figs. 15 to 17 show the total-order Sobol' indices for these UT cases. The black lines in these plots show the standard deviations due to the ten different sample sets chosen. For Case 1, the F-number has negligible effect on the model response as seen in Figs. 12 and 15, respectively. The same is true for Case 3, shown in Figs. 14 and 17, respectively. In Case 2, the y-coordinates has small effect on the model response as seen in

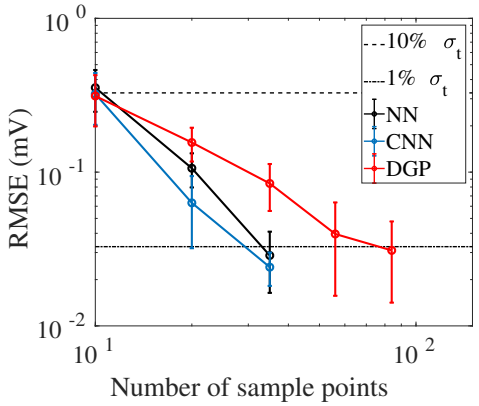

Fig. 6. RMSE for Case 1 at a = 0.5 mm

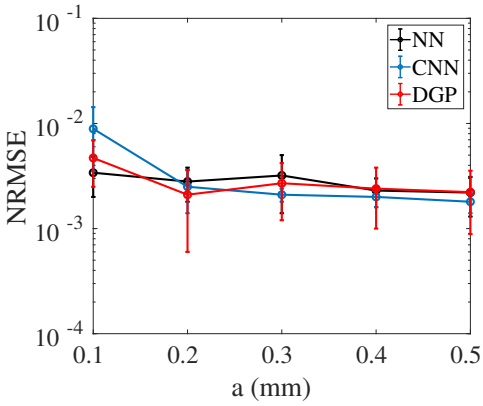

Fig. 9. NRMSE for Case 1

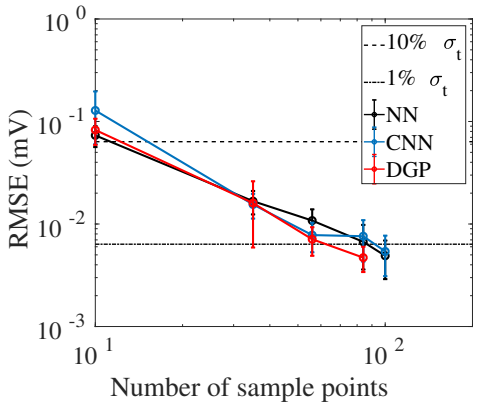

Fig. 7. RMSE for Case 2 at a = 0.5 mm

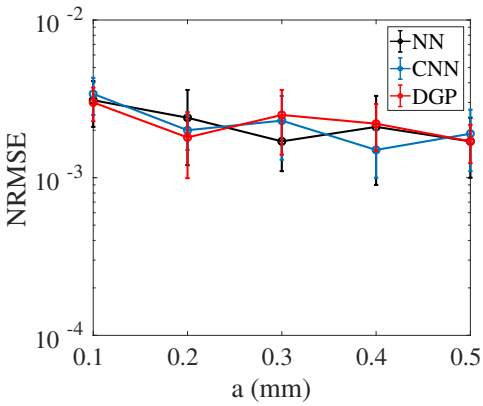

Fig. 10. NRMSE for Case 2

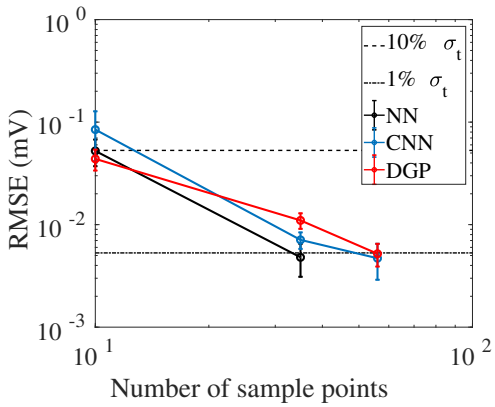

Fig. 8. RMSE for Case 3 at a = 0.5 mm

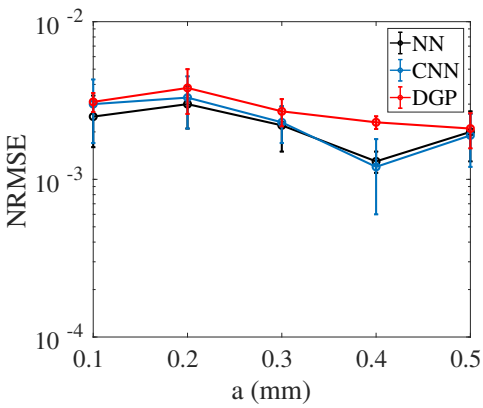

Fig. 11. NRMSE for Case 3

Figs. 13 and 16, respectively. For all the different cases, the Sobol' indices values from the machine learning algorithms match well with those from the physics-based models.

4 Conclusion

Three different machine learning algorithms, namely, neural networks, convolutional neural networks, and deep Gaussian processes, were used to perform model-based sensitivity analysis for ultrasonic testing systems using three benchmark cases developed by the WFNDEC. First, the global accuracy of these algorithms were measured and the number of high-fidelity samples required to reach the desired global accuracy were noted. These globally accurate algorithms were then used to generate model responses in order to perform sensitivity analysis using Sobol' indices. The sensitivity analysis results also matched well with those obtained by directly using the physics-based model.

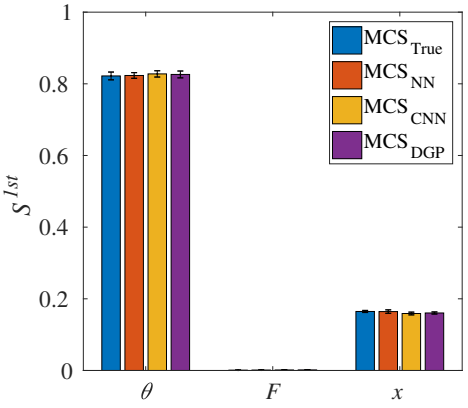

Fig. 12. 1st-order Sobol' indices for Case 1

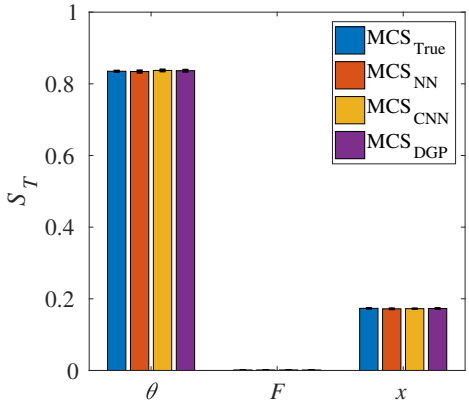

Fig. 15. Total-order Sobol' indices for Case 1

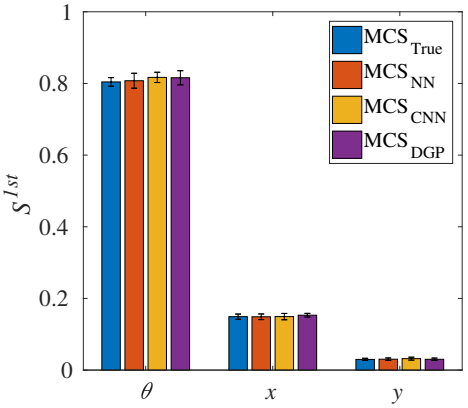

Fig. 13. 1st-order Sobol' indices for Case 2

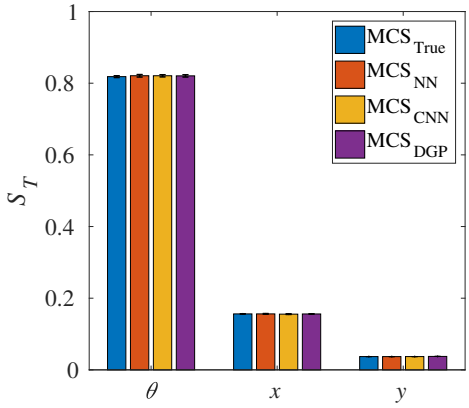

Fig. 16. Total-order Sobol' indices for Case 2

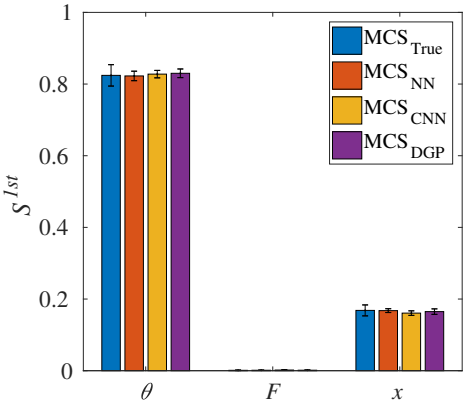

Fig. 14. 1st-order Sobol' indices for Case 3

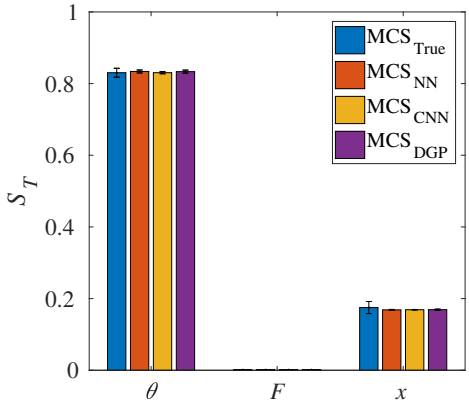

Fig. 17. Total-order Sobol' indices for Case 3

This study shows that NN, CNN, and DGP machine learning algorithms can be used to provide fast and accurate sensitivity results values. Performing sensitivity analysis can assist in deciding which variability parameters need to be considered while performing physical experiments, which can reduce both cost and time of the experiments. Future work will include cases with non-spherical defects as well as cases with a larger number of variability parameters.

Acknowledgements

The authors are supported in part by NSF award number 1846862, and the Iowa State University Center for Nondestructive Evaluation Industry-University Research Program.

References

[1] Cawley, P., 2001. "Non-Destructive Testing - Current Capabilities and Future Directions". *Journal of Materials: Design and Applications,* **215**, pp. 213–223.

[2] Sharma, A., and A., S. K., 2018. "Ultrasonic Testing for Mechanical Engineering Domain: Present and Future Perspective ". *International Journal of Research in Industrial Engineering,* **7**(2), pp. 243–253.

[3] Capriotti, M., Kim, E. H., Scalea, L. F., and Kim, H., 2017. "Non-Destructive Inspection of Impact Damage in Composite Aircraft Panels by Ultrasonic Guided Waves and Statistical Processing". *Materials,* **10**(6), pp. 1–12.

[4] Poudel, A., Strycek, J., and Chu, T. P., 2013. "Air-Coupled Ultrasonic Testing of Carbon-Carbon Composite Aircraft Brake Disks". *Materials Evaluation,* **71**, pp. 987–994.

[5] Choi, S.-N., Hong, S.-Y., and Hwang, W.-G., 2012. "Performance Demonstration for an Automated Ultrasonic Testing System for Piping Welds". *Journal of Nuclear Sience and Technology,* **49**, pp. 562–570.

[6] Adamus, K., and Lacki, P., 2017. "Assessment of Aluminum FSW Joints Using Ultrasonic Testing". *Archives of Metallurgy and Materials,* **62**, pp. 2399–2404.

[7] Bai, L., Velichko, A., and Drinkwater, B. W., 2018. "Ultrasonic Defect Characterization - Use of Amplitude, Phase and Frequency Information". *The Journal of the Acoustic Society of America,* **143**, pp. 349–360.

[8] Ahmed, A., Badarinarayan, K. S., R., N. A., and G., G., 2015. "Development of Ultrasonic Reference Standards for Defect Characterization in Carbon Fiber Composites". *International Research Journal of Engineering and Technology (IRJET),* **2**, pp. 840–844.

[9] Ginzel, E., 2007. NDT Modelling An Overview. Technical report, Materials Research Institute, Waterloo, Ontario, Canada.

[10] Darmon, M., Chatillon, S., Mahaut, S., Calmon, P., Fradkin, L., and Zernov, V., 2011. "Recent Advances in Semi-Analytical Scattering Models for NDT Simulation". *Journal of Physics,* **269**, pp. 1–12.

[11] Kolkoori, S., 2014. "Quantitative Evaluation of Ultrasonic Wave Propagation in Inhomogeneous Anisotropic Austenitic Welds Using 3D Ray Tracing Method: Numerical and Experimental Validation". PhD Thesis, Technical University Berlin, Berlin, Germany, April.

[12] Wagner, D., Cavalieri, F. J., Bathias, C., and Ranc, N., 2012. "Ultrasonic Fatigue Tests at High Temperature on an Austenitic Steel". *Propulsion and Power Research,* **1**, pp. 29–35.

[13] Subair, S., Balasubramaniam, K., Rajagopal, P., Kumar, A., Rao, B. P., and Jayakumar, T., 2014. "Finite Element Simulations to Predict Probability of Detection (PoD) Curves for Ultrasonic Inspection of Nuclear Components". *Procedia Engineering,* **86**, pp. 461–468.

[14] Zhang, C., and Gross, D., 2002. "A 2D Hyper Singular Time-Domain Traction BEM for Transient Elastodynamic Crack Analysis". *Wave Motion,* **35**, pp. 17–40.

[15] Westlund, J., 2011. "On the Propagation of Ultrasonic Testing Using Boundary Integral Equation Method". PhD Thesis, Chalmers University of Technology, Gothenburg, Sweden, January.

[16] Jeong, H., and Schmerr, L. W., 2007. "Ultrasonic Beam Propagation in Highly Anisotropic Materials Simulated by Multi Gaussian Beams". *Journal of Mechanical Science and Technology,* **21**, pp. 1184–1190.

[17] Ye, J., Kim, H. J., Song, S. J., Kang, S. S., Kim, K., and Song, M. H., 2011. "Model Based Simulation of Focused Beam Fields Produced by a Phased Array Ultrasonic Transducer in Dissimilar Meta Welds". *NDT/E International,* **44**, pp. 290–296.

[18] Nam, Y.-H., 2001. "Modeling of Ultrasonic Testing in Butt Joint by Ray Tracing". *Journal of Mechanical Science and Technology,* **15**, pp. 441–447.

[19] Liu, Q., Persson, G., and Wirdelius, H., 2014. "A Receiver Model for Ultrasonic Ray Tracing in an Inhomogeneous Anistropic Weld". *Journal of Modern Physics,* **5**, pp. 1186–1201.

[20] Ferretti, F., Saltelli, A., and Tarantola, S., 2016. "Trends in Sensitivity Analysis Practice in the Last Decades". *Science of the Total Environment,* **568**, pp. 666–670.

[21] Staelen, R. H. D., and Beddek, K., 2015. "Sensitivity Analysis and Variance Reduction in a Stochastic NDT Problem". *International Journal of Computer Mathematics,* **92**, pp. 1874–1882.

[22] Ghanem, R., Higdon, D., and Owhadi, H., eds., 2017. *Handbook of Uncertainty Quantification*, 1^{st} ed. Springer International Publishing, Switzerland, pp. 1–20.

[23] Sher, A., Wang, K., Wathen, A., Maybank, P., Mirams, G., Abramson, D., Noble, D., and Gavaghan, D., 2011. "A Local Sensitivity Analysis Method for Developing Biological Models with Identifiable Parameters: Application to Cardiac Ionic Channel Modelling". *Future Generation Computer System,* **29**, pp. 591–598.

[24] Homma, T., and Saltelli, A., 1996. "Importance Measures in Global Sensitivity Analysis of Nonlinear Models". *Reliability Engineering and System Safety,* **52**, pp. 1–17.

[25] Morio, J., 2011. "Global and Local Sensitivity Analysis Methods for a Physical System". *European Journal of Physics,* **32**, pp. 1–9.

[26] Sobol', I., and Kucherekoand, S., 1993. "Sensitivity estimates for nonlinear mathematical models". *Mathematical Modelling and Computational Experiments,* **1**, pp. 407–414.

[27] Sobol', I., 2001. "Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates". *Mathematics and Computers in Simulation,* **55**, pp. 271–280.

[28] Forrester, A. I. J., and Keane, A. J., 2009. "Recent Advances in Surrogate-Based Optimization". *Progress in Aerospace Sciences,* **45**(1-3), pp. 50–79.

[29] Queipo, N. V., Haftka, R. T., Shyy, W., Goel, T., Vaidyanathan, R., and Tucker, P. K., 2005. "Surrogate-Based Analysis and Optimization". *Progress in Aerospace Sciences,* **41**(1), pp. 1–28.

[30] Forrester, A., Sobester, A., and Keane, A., 2008. *Engineering Design via Surrogate Modelling: A Practical Guide*. John Wiley and Sons, Ltd., United Kingdom.

[31] Peherstorfer, B., Willcox, K., and Gunzburger, M., 2018. "Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization". *Society for Industrial and Applied Mathematics,* **60**(3), pp. 550–591.

[32] Forrester, I. J. A., Sobester, A., and Keane, J. A., 2007. "Multi-Fidelity Optimization via Surrogate Modelling". *Proceedings of the Royal Society A Mathematical Physical and Engineering Sciences,* **463**(2088), pp. 3251–3269.

[33] Wiener, N., 1938. "The Homogeneous Chaos". *American Journal of Mathematics,* **60**(4), pp. 897–936.

[34] Krige, D. G., 1951. "A Statistical Approach to Some Basic Mine Valuation Problems on the Witwatersrand". *Journal of the Chemical, Metallurgical and Mining Engineering Society of South Africa,* **52**(6), pp. 119–139.

[35] Li, D., Wilson, P. A., and Jiong, Z., 2015. "An Improved Support Vector Regression and Its Modelling of Manoeuvring Performance in Multidisciplinary Ship Design Optimization". *International Journal of Modelling and Simulation,* **35**, pp. 122–128.

[36] Kennedy, C. M., and O'Hagan, A., 2000. "Predicting the Output from a Complex Computer Code When Fast Approximations are available". *Biometrika,* **87**(1), pp. 1–13.

[37] Echeverria, D., and Hemker, P., 2008. "Manifold mapping: A two-level optimization technique". *Computing and Visualization in Science,* **11**, pp. 193–206.

[38] Bilicz, S., Vazquez, E., Gyimothy, S., Pavo, J., and Lambert, M., 2010. "Kriging for Eddy-Current Testing Problems". *IEEE Transactions on Magnetics,* **46**, pp. 4582–4590.

[39] Bilicz, S., Lambert, M., Gyimothy, S., and Pavo, J., 2012. "Solution of Inverse Problems in Nondestructive Testing by a Kriging-Based Surrogate Model". *IEEE Transactions on Magnetics,* **48**, pp. 495–498.

[40] Miorelli, R., Artusi, X., Addessalem, B. A., and Reboud, C., 2016. "Database Generation and Exploitation for Efficient and Intensive Simulation Studies". *Review of Progress in Quantitative Nondestructive Evaluation,* **1706**, pp. 180002–1–180002–9.

[41] Du, X., Leifsson, L., Meeker, W., Gurrala, P., Song, J., and Roberts, R., 2019. "Efficient Model-Assisted Probability of Detection and Sensitivity Analysis for Ultrasonic Testing Simulations Using Stochastic Metamodeling". *Journal of Nondestructive Evaluation, Diagnostics and Prognostics of Engineering Systems,* **2(4): 041002**(4), 09.

[42] Blatman, G., 2009. "Adaptive Sparse Polynomial Chaos Expansion for Uncertainty Propagation and Sensitivity Analysis". *Ph.D. Thesis, Blaise Pascal University - Clermont II. 3, 8, 9*.

[43] Du, X., and Leifsson, L., 2020. "Multifidelity Modeling by Polynomial Chaos-Based Cokriging to Enable Efficient Model-Based Reliability Analysis of NDT Systems". *Journal of Nondestructive Evaluation,* **39(3)**.

[44] Schmidhuber, J., 2015. "Deep learning in neural networks: An overview". *Neural Networks,* **61**, pp. 85 – 117.

[45] Haykin, S. S., 2009. *Neural networks and learning machines*, 3^{rd} ed. Pearson Education, Upper Saddle River, NJ.

[46] Goodfellow, I., Bengio, Y., and Courville, A., 2016. *Deep Learning*. The MIT Press, Cambridge, MA.

[47] le Cun, Y., 1989. Generalization of Network Design Strategies. Technical report, University of Toronto, Toronto, Canada, June.

[48] Damianou, A., and Lawrence, N., 2013. "Deep Gaussian processes". In Artificial Intelligence and Statistics, C. M. Carvalho and P. Ravikumar, eds., Vol. **31**, PMLR, pp. 207–215.

[49] Salimbeni, H., and Deisenroth, M., 2017. "Doubly stochastic variational inference for deep gaussian processes". In Advances in Neural Information Processing Systems 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., Curran Associates, Inc., pp. 4588–4599.

[50] Thompson, R. B., and Gray, T. A., 1983. "Analytic Diffraction Corrections to Ultrasonic Scattering Measurements". *Library of Congress Cataloging in Publication Data, Springer,* **2A**.

[51] Schmerr, L., and Song, J., 2007. *Ultrasonic Nondestructive Evaluation Systems*. Springer Science + Business Media, LLC, New York, USA.

[52] McKay, M. D., Beckman, R. J., and Conover, W. J., 1979. "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code". *Technometrics,* **21**(2), pp. 239–245.

[53] Metropolis, N., and Ulam, S., 1949. "The Monte Carlo Method". *Journal of the American Statistical Association,* **44**(247), pp. 335–341.

[54] Garg, V. V., and Stogner, R. H., 2017. "Hierarchical Latin Hypercube Sampling". *Journal of the American Statistical Association,* **112**(518), pp. 673–682.

[55] Chauvin, Y., and Rumelhart, D. E., 1995. *Backpropagation: theory, architectures, and applications*. Psychology press, Hillsdale, NJ.

[56] Kingma, D. P., and Ba, J., 2014. Adam: A method for stochastic optimization. arXiv:1412.6980.

[57] Chollet, F., et al., 2015. Keras. https://keras.io.

[58] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P.,

Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[59] Ryu, J., Kim, K., Lee, T., and Choi, D., 2002. "Kriging Interpolation Methods in Geostatistics and DACE Model". *Korean Society of Mechanical Engineering International Journal,* **16**(5), pp. 619–632.

[60] Gneiting, T., Kleiber, W., and Schlather, M., 2010. "Matern Cross-Covariance Functions for Multivariate Random Fields". *Journal of the American Statistical Association,* **105**(491), pp. 1167–1177.