A Machine Learning Framework for Reduced Order Modeling of Guided Waves Propagation

George Drakoulas[1,2], Theodoros Gortsas[1], Charilaos Kokkinos[2], Fotis Kopsaftopoulos[3] and Demosthenes Polyzos[1]

[1]Department of Mechanical Engineering and Aeronautics, University of Patras, GR-26500, Rion, Greece

[2]FEAC Engineering P.C., GR-26224, Patras, Greece

[3]Department of Mechanical, Aerospace, and Nuclear Engineering, Rensselaer Polytechnic Institute, NY 12180, Troy, USA

Abstract. In structural health monitoring, it is critical to quantify the uncertainty in the knowledge of the internal parameters, which influence the response of a mechanical system to external loading. In thin structures, the variation of the mechanical properties, influences significantly the guided wave propagation taking place when dynamical loads are applied. Computational structural mechanics is a powerful tool to accurately evaluate the impact of the uncertainty in the material properties, to the mechanical behavior of thin structures. To reduce the computational cost and provide numerical solutions in a limited amount of time, the high fidelity computational models are replaced by reduced-order models (ROMs). This paper utilizes a state-of-the-art, machine learning (ML)-based ROM methodology combined with a singular value decomposition (SVD) update algorithm to reduce the simulation time. The developed framework, mentioned as *FastSVD-ML-ROM*, includes both linear and non-liner dimensionality reduction techniques by using SVD and convolutional autoencoders to extract latent variables. Fully connected neural networks are utilised to map the input parameter sets to the latent variables and long-short term memory networks, to predict the temporal scales. To evaluate the performance of *FastSVD-ML-ROM* the guided wave propagation taking place in a lightweight aluminum plate is examined. The numerical data needed to train and test *FastSVD-ML-ROM*, is obtained with simulations performed with the finite element method by varying the young modulus, poisson ratio, and the density of the plate. The accuracy of the predicted solutions showcases the robustness of the *FastSVD-ML-ROM* and demonstrates the ability to quantify the uncertainties introduced in the material properties of the propagating medium by significantly reducing the computational cost.

Keywords— Reduced order modeling, Uncertainty quantification, Wave propagation, Structural, health monitoring, Machine learning

1 Introduction

Structural health monitoring (SHM) is a commonly applied technology in the field of civil, aerospace, and mechanical engineering, providing great insight on the structural performance of components, which in turn helps to increase their reliability and safety [1]. SHM methods target to monitor the relationship between the variations introduced into the system parameters (e.g. material properties, boundary conditions, etc.) and the operational performance mainly through the installation of distributed sensors (e.g. accelerometers) [2]. However, the monitoring of the structural behavior becomes challenging when dealing with uncertainties involved in the propagation of guided waves, over the variations in the material properties [3].

Computational structural mechanics has proven to be a unique tool in the development of numerical models. High fidelity models (HFM) can precisely represent the performance of the real system, and provide a useful insight before the manufacturing phase [4]. To predict the physical behavior of the structures, numerical schemes are commonly utilized. The finite element method (FEM) and the boundary element method (BEM) demonstrate accurate results when solving large-scale and transient structural dynamics problems [5]. Consequently, computational structural mechanics in terms of numerical analysis is a useful and accurate methodology, for the simulation of complex physical phenomena, including guided wave propagation problems [6]. However, the computational cost is rapidly increasing when the sensitivity of the structure to the variations of the material properties needs to be

determined. In particular, multiple numerical simulations are required to estimate the uncertainty quantification [7, 8]. In that frame, high performance computing is often essential.

At the same time, the demand for multi-fidelity approaches consisting of simplified techniques, able to deliver results in a limited amount of time, is rapidly increasing [9]. Along these lines, surrogate modeling has gained attention, aiming to reduce the computational cost of HFMs while providing results in a limited amount of time [10]. Reduced-order modeling (ROM), as part of surrogate modeling, has extensively been applied to decrease the simulation cost, while preserving the dominant effects of the HFM [11]. ROMs can be classified into two main categories according to the availability of the underlying partial differential equations that describe the physical phenomenon: *i)* intrusive ROMs (IROMs) and *ii)* non-intrusive ROMs (NIROMs).

Regarding IROMs, the exact form of the underlying partial differential equations is essential to generate simplified models from high fidelity solutions [12]. Proper orthogonal decomposition is utilised to approximate the intrinsic coordinates of the HFM, and through galerkin projection, predict the evolution of the dynamics [12]. However, IROMs do not maintain stability and provide inaccurate results when predicting unsteady, non-linear cases. Moreover, they require access to the numerical solvers, which is impossible when using proprietary software [13]. In recent years, NIROMs DL-based have been the subject of several studies by recovering non-linear, low-dimensional manifolds and predict the dynamics of non-linear partial differential equations (PDEs) [14, 15, 16]. A common methodology applied in the frame of NIROMs DL-based is the use of convolutional autoencoders (CAE) for the non-linear dimensionality reduction, and the long short-term memory (LSMT) networks for the temporal evolution [17].

In this work, a NIROM methodology is utilised to approximate the high fidelity solutions of the wave propagation, extracted by structural analysis. The main goal is to set-up an artificial intelligence platform able to perform a forward uncertainty quantification (UQ) analysis and to examine the effects of the variation in the material properties. To achieve that, HFMs are simulated using the FEM aiming to provide the unsteady acceleration distribution through the uncertainties of the young modulus, the poisson's ratio, and the density introduced in a light-weight aluminum plate. As a second step, the NIROM is trained and tested by the generated numerical data. Finally, the developed NIROM is used to replace the HFM and investigate the UQ of the acceleration fields by simulating several cases, while reducing the computational cost several orders of magnitude.

We utilize a state-of-the-art ML-NIROM framework [18], that includes *i)* a singular value decomposition (SVD) update methodology to calculate a linear subspace of FOM during the simulation process, *ii)* CAE for the non-linear dimensionality reduction, *iii)* a feed-forward neural network (FFNN) to map the input parameters to the latent spaces extracted through CAEs and *iv)* LSTM networks to predict and forecast the dynamics of the parametric solutions. The used framework, referred to as *FastSVD-ML-ROM* can handle large-scale FOMs, and, calculate the linear embedding during the simulation process, while identifying non-linear manifolds.

The structure of this paper is as follows. Section 2, presents the methodology for the snapshot matrix construction, the SVD update methodology, and the CAEs for the non-linear dimensionality reduction, as well. Moreover, the methodology for the prediction of the dynamics formulated as a combination of FFNN and LSTMs and the approach followed for the UQ, are described. Then in section 3, a description of the FastSVD-ML-ROM is presented in detail, and the integration of the FastSVD-ML-ROM with the UQ methodology is described. Then in section 4, the implementation of the developed framework for the UQ of guided waves propagation through the variation of the material properties, is presented. Finally, a brief discussion of the results is described in section 5.

2 Machine learning-based reduced-order modeling

This section describes the setting of the transient numerical problem as well as the basic components of the employed *FastSVD-ML-ROM* framework.

2.1 Problem setting

The general form of a time-dependent, parameterised, PDE in structural mechanics can be expressed as follows:

$$\ddot{\boldsymbol{u}}(t,\boldsymbol{x};\boldsymbol{\mu}) = \mathscr{F}(\boldsymbol{x};\boldsymbol{u};\boldsymbol{\mu}),\ \ \dot{\boldsymbol{u}}(0,\boldsymbol{x};\boldsymbol{\mu}) = \dot{\boldsymbol{u}}_0(\boldsymbol{x};\boldsymbol{\mu}),\ \ \boldsymbol{u}(0,\boldsymbol{x};\boldsymbol{\mu}) = \boldsymbol{u}_0(\boldsymbol{x};\boldsymbol{\mu}),\ \ t \in (0,T],\ \ \boldsymbol{x} \in \Omega \tag{1}$$

where, $\Omega \subset \mathbb{R}^d$ is the spatial domain of interest, $\boldsymbol{u}$: $\mathbb{R}^+ \times \Omega \times \mathbb{R}^\xi \rightarrow \mathbb{R}^g$ is the unknown displacement vector, $\mathscr{F}$ is a linear differential operator, $\boldsymbol{u}_0$: $\mathbb{R}^d \times \mathbb{R}^\xi \rightarrow \mathbb{R}^g$ and $\dot{\boldsymbol{u}}_0$: $\mathbb{R}^d \times \mathbb{R}^\xi \rightarrow \mathbb{R}^g$ are the initial conditions, $\boldsymbol{x} \in \Omega$ denotes a spatial point, $(0,T]$ is the temporal domain, ξ is the dimension of the parameter vector and d is the number of spatial dimensions. The well-posedness of the problem (Eq. 1), requires a certain number of boundary conditions, that must be satisfied to have unique solutions on $\partial\Omega$. The parameter vector $\boldsymbol{\mu}$ is sampled from a parameter space $\mathscr{P} \subset \mathbb{R}^\xi$, that constitutes the space of interest for the HFM solution, and it can consist for example of boundary or initial conditions and material properties amongst others.

The discretized form of Eq. 1 can be represented in the following form,

$$\ddot{\boldsymbol{u}}_h(t;\boldsymbol{\mu}) = \mathscr{F}_h(\boldsymbol{u}_h;\boldsymbol{\mu}),\ \ \dot{\boldsymbol{u}}_h(0;\boldsymbol{\mu}) = \dot{\boldsymbol{u}}_h(\boldsymbol{\mu}),\ \ \boldsymbol{u}_h(0;\boldsymbol{\mu}) = \boldsymbol{u}_{\boldsymbol{h}}(\boldsymbol{\mu}),\ \ t \in (0,T] \tag{2}$$

where $h > 0$ denotes a discretization parameter, such as the element size. The FOM parameterised discrete solution is denoted as $\boldsymbol{u}_h(t;\boldsymbol{\mu})$: $\mathbb{R}^+ \times \mathbb{R}^\xi \rightarrow \mathbb{R}^{N_h}$, and N_h the dimension of the discretized function $\boldsymbol{u}$. The $\boldsymbol{\mu}$ parameters are sampled from the space

$\mathscr{P}$ using a probability distribution. They are splitted in two sets $\boldsymbol{\mu}^{tr}$=$\{\boldsymbol{\mu}_1^{tr},\ldots,\boldsymbol{\mu}_m^{tr}\}$ and $\boldsymbol{\mu}^{te}$=$\{\boldsymbol{\mu}_1^{te},\ldots,\boldsymbol{\mu}_n^{te}\}$ corresponding to the m training and n testing parameters, respectively, where $\mu^{tr} \cup \mu^{te} \subset \mathscr{P}$. The snapshot matrix $\boldsymbol{S}_h$ is defined for the parameter set μ^{tr} and the discrete time set $\{t_1,\ldots,t_{N_t}\}$ with $t_i \in (0,T]$, as follows,

$$\boldsymbol{S}_h = [\, \boldsymbol{u}_h(t_1;\boldsymbol{\mu}_1^{tr}) \,|\ldots|\, \boldsymbol{u}_h(t_{N_t},\boldsymbol{\mu}_1^{tr}) \,|\ldots|\, \boldsymbol{u}_h(t_1,\boldsymbol{\mu}_m^{tr}) \,|\ldots|\, \boldsymbol{u}_h(t_{N_t},\boldsymbol{\mu}_m^{tr}) \,] \tag{3}$$

2.2 Dimensionality reduction

The *FastSVD-ML-ROM* uses a hyper-reduction technique based on both linear algebra (SVD) and CAE to efficiently compress the snapshot matrix extracted through Eq. 3. Primarily, a truncated SVD update [18] is employed during the simulation process to obtain an approximate basis vector $\boldsymbol{U}$.

Using the calculated basis the reduced data is obtained as,

$$\tilde{\boldsymbol{u}}_h(t_i;\boldsymbol{\mu}_j^{tr}) \approx \boldsymbol{U}\ \boldsymbol{U}^T \boldsymbol{u}_h(t_i;\boldsymbol{\mu}_j^{tr}) \tag{4}$$

Given the snapshot matrix containing the solutions for all the parameters, we calculate the following low-dimensional representation,

$$\boldsymbol{S}_n = \boldsymbol{U}^T \boldsymbol{S}_h \tag{5}$$

where the projected matrix $\boldsymbol{S}_n$ is the input of the CAE and n the column dimensions of the basis vector $\boldsymbol{U}$ derived during the SVD update. Therefore, the projected FOM solution for the training parameter $\boldsymbol{\mu}_j^{tr}$ at time t_i is derived though the formula,

$$\boldsymbol{u_n}(t_i,\boldsymbol{\mu}_j^{tr}) = \boldsymbol{U}^T \boldsymbol{u_h}(t_i;\boldsymbol{\mu}_j^{tr}) \tag{6}$$

As a second step, the projected solutions $\boldsymbol{u_n}(t_i,\boldsymbol{\mu}_j^{tr})$, with j=1,…,m and i=1,…,N_t, are given as input to the CAE, aiming to identify a non-linear, low dimensional, latent variables.

In particular, the CAE maps the input tensor to a latent space through the encoder denoted as e. This is achieved by using convolutional layers, max-pooling operations and a FFNN. In the inverse direction, the decoder denoted as d transforms the reduced solutions following a similar architecture, by converting the latent spaces in a vector through a FFNN, followed by multiple convolutional layers and upsampling layers to generate the reconstructed output. A detailed explanation of the CAE architecture is presented in [15].

Let $\boldsymbol{\phi}_e$ be the encoder function that attempts to discover a latent representation $\tilde{\mathbf{z}}_q \in \mathbb{R}^q$, with $q << n$,

$$\tilde{\mathbf{z}}_{\boldsymbol{q}}(t_i;\boldsymbol{\mu}_j^{tr}) = \boldsymbol{\phi}_e(\boldsymbol{u}_n(t_i;\boldsymbol{\mu}_j^{tr});\boldsymbol{W}_e;\boldsymbol{b}_e),\ \ with\ \ i=1,\ldots,N_t\ \ and\ \ j=1,\ldots,m \tag{7}$$

while $\boldsymbol{W}_e$, $\boldsymbol{b}_e$ are groups of weight matrices and bias vectors of the encoder, respectively.

The decoder reconstructs the approximate input data $\tilde{\boldsymbol{u}}_n(t_i;\boldsymbol{\mu}_j^{tr})$ through the following transformation,

$$\tilde{\boldsymbol{u}}_n(t_i;\boldsymbol{\mu}_j^{tr}) = \boldsymbol{\phi}_d(\tilde{\mathbf{z}}_q(t_i;\boldsymbol{\mu}_j^{tr});\boldsymbol{W}_d;\boldsymbol{b}_d) = \boldsymbol{\phi}_d(\boldsymbol{\phi}_e(\boldsymbol{u}_n(t_i;\boldsymbol{\mu}_j^{tr});\boldsymbol{W}_e;\boldsymbol{b}_e);\boldsymbol{W}_d;\boldsymbol{b}_d) \tag{8}$$

where the transition functions are defined as $\boldsymbol{\phi}_e : \mathbb{R}^n \to \mathbb{R}^q$ and $\boldsymbol{\phi}_d : \mathbb{R}^q \to \mathbb{R}^n$, while $\boldsymbol{W}_d$, $\boldsymbol{b}_d$ are the weight matrices and bias vectors of the decoder.

2.3 Time predictions

In the frame of *FastSVD-ML-ROM*, to predict the spatio-temporal solutions of the reduced data $\tilde{\mathbf{z}}_q$, which is extracted from the dimensionality reduction (Section 2.2), a FFNN and LSTM network is utilised. In particular, the DL-models are trained independently over the training and then during the online phase, the FFNN predicts and provides the first time series to the LSTM network, used to forecast the temporal solutions.

Initially, LSTMs are used to predict the evolution of the latent variables of the FOM snapshots. We denote the LSTM network, as l that takes as input the vector $\boldsymbol{x}_l^i(\boldsymbol{\mu}_j^{tr})$, consisting of the latent spaces $\tilde{\mathbf{z}}_q(t_i,\boldsymbol{\mu}_j)$ with $i=1,\ldots,N_t$ and $j=1,\ldots,m$ extracted through the trained encoder $\boldsymbol{\varphi}_e$. To explicitly account for the parameter information, we concatenate the latent spaces with the parameter values, and thus $\boldsymbol{x}_l^i(\boldsymbol{\mu}_j^{tr}) = [\tilde{\mathbf{z}}_q(t_i;\boldsymbol{\mu}_j^{tr}),\ \boldsymbol{\mu}_j^{tr}]$.

In particular, multiple samples are generated through a predefined sliding window w, arranged in a temporal sequence to enter the LSTM network. Each input set is passed incrementally for each parameter $\boldsymbol{\mu}_j^{tr}$ has the following form,

$$\mathscr{X}_l(\boldsymbol{\mu}_j^{tr}) = \{\ \boldsymbol{X}_l^1(\boldsymbol{\mu}_j^{tr}),\ldots,\boldsymbol{X}_l^g(\boldsymbol{\mu}_j^{tr})\ \} \tag{9}$$

where $g=(N_t-w)$, and each matrix $\boldsymbol{X}_l^i(\boldsymbol{\mu}_j^{tr})$ contains w column vectors, given as,

$$\boldsymbol{X}_l^i(\boldsymbol{\mu}_j^{tr}) = [\, \boldsymbol{x}_l^i(\boldsymbol{\mu}_j^{tr}) \,|\ldots|\, \boldsymbol{x}_l^{(i+w-1)}(\boldsymbol{\mu}_j^{tr}) \,] \tag{10}$$

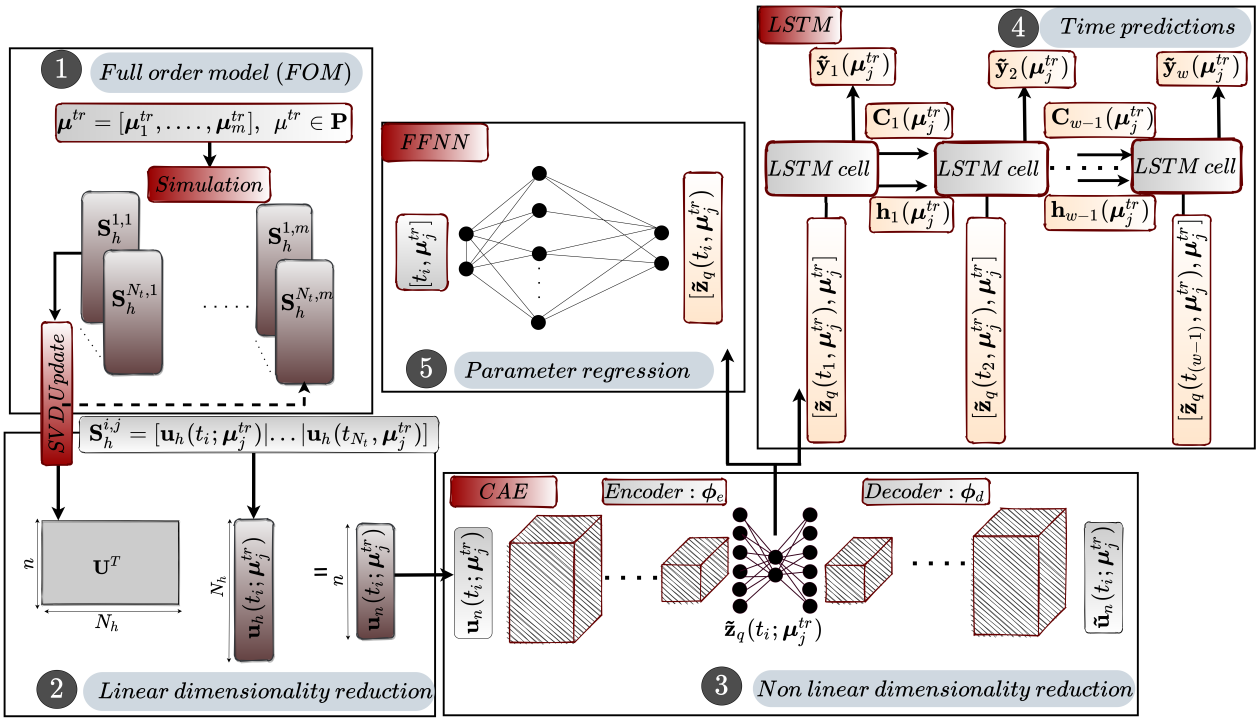

Fig. 1: Offline phase (training) of the *FastSVD-ML-ROM*.

The LSTM network is trained by the data $\boldsymbol{X}_l^i(\boldsymbol{\mu}_j^{tr})$ to predict the next latent space vector $\boldsymbol{y}_l^i(\boldsymbol{\mu}_j^{tr}) = \tilde{\mathbf{z}}_\mathbf{q}(t_{i+w}; \boldsymbol{\mu}_j^{tr})$. Given that the LSTM model is deployed for all matrices $\boldsymbol{X}_l^i(\boldsymbol{\mu}_j^{tr})$, the outputs of the training sequence for each parameter $\boldsymbol{\mu}_j^{tr}$ form the following set $\mathscr{Y}_l$,

$$\mathscr{Y}_l(\boldsymbol{\mu}_j^{tr}) = \{\boldsymbol{y}_l^1(\boldsymbol{\mu}_j^{tr}), \ldots, \boldsymbol{y}_l^g(\boldsymbol{\mu}_j^{tr})\} \tag{11}$$

The LSTM network is applied in the sequence presented for each training parameter set $\boldsymbol{\mu}_j^{tr}$ at time t_i and mainly consists of three gates acting as filters to the input data. The forget gate $\boldsymbol{f}_i(\boldsymbol{\mu}_j^{tr})$ that decides which information is less important and can be ignored, the input gate $\boldsymbol{i}_i(\boldsymbol{\mu}_j^{tr})$ used to update the cell state, and the output gate $\boldsymbol{o}_i(\boldsymbol{\mu}_j^{tr})$ determines the information that will be passed to the next state. Besides, the update cell state $\boldsymbol{C}_i(\boldsymbol{\mu}_j^{tr})$, composed by the input $\boldsymbol{i}_i(\boldsymbol{\mu}_j^{tr})$, the forget gate $\boldsymbol{f}_i(\boldsymbol{\mu}_j^{tr})$ and the candidate memory state $\tilde{\boldsymbol{C}}_i(\boldsymbol{\mu}_j^{tr})$ generates data. The hidden state $\boldsymbol{h}_i(\boldsymbol{\mu}_j^{tr})$ produces the output data given the filtered data from the output gate $\boldsymbol{o}_i(\boldsymbol{\mu}_j^{tr})$ [19]. Then, the hidden state is passed to a FFNN layer aiming to predict the final output $\tilde{\boldsymbol{y}}_l^1(\boldsymbol{\mu}_j^{tr})$. A detailed explanation of the LSTM architecture can be found in [18].

We denote as $\boldsymbol{\varphi}_l$ the general function of each LSTM cell that attempts to predict the next step given incrementally the input data $\boldsymbol{X}_l^i(\boldsymbol{\mu}_j^{tr})$,

$$\tilde{\mathbf{y}}_l^i(\boldsymbol{\mu}_j^{tr}) = \boldsymbol{\varphi}_l(\boldsymbol{X}_l^i(\boldsymbol{\mu}_j^{tr}); \boldsymbol{W}_l; \boldsymbol{b}_l) \tag{12}$$

where $i = 1, \ldots, g$ and $j = 1, \ldots, m$.

During the offline phase, along with the training of the LSTMs, FFNNs are used to identify the latent space vectors $\tilde{z}_\mathbf{q}(t_i; \boldsymbol{\mu}_j^{tr})$ for $i = 1, \ldots, w$ and $j = 1, \ldots, m$ that form the first time series matrix $\mathbf{X}_l^1(\boldsymbol{\mu}_j^{tr})$, $j = 1, \ldots, m$, provided as input to the LSTM during the online phase to predict and forecast the dynamics.

In particular, FFNNs, also known as multi-layer perceptrons, approximate the functional relation amongst a set of input and output values.

To formulate the input for the FFNNs, we concatenate the time instances t_i, $i = 1, \ldots, w$ with the parameters $\boldsymbol{\mu}_j^{tr}$, and thus $\boldsymbol{x}_f^i(\boldsymbol{\mu}_j^{tr}) = [t_i, \boldsymbol{\mu}_j^{tr}]$. In particular the training set of the FFNN is the following,

$$\boldsymbol{x}_f(\mu_j^{tr}) = \{\boldsymbol{x}_f^1(\mu_j^{tr}), \ldots, \boldsymbol{x}_f^w(\mu_j^{tr})\} \tag{13}$$

The output of the FFNN for a given parameter $\boldsymbol{\mu}_j^{tr}$ is the following set

$$\boldsymbol{y}_f(\mu_j^{tr}) = \{\boldsymbol{y}_f^i(\boldsymbol{\mu}_j^{tr}), \ldots, \boldsymbol{y}_f^w(\boldsymbol{\mu}_j^{tr})\} \tag{14}$$

where, $\boldsymbol{y}_f^i(\boldsymbol{\mu}_j^{tr}) = \tilde{z}_{\boldsymbol{q}}(t_i;\boldsymbol{\mu}_j^{tr})$. We aim to obtain the nonlinear mapping such that:

$$\tilde{\mathbf{y}}_f^i(\mu_j^{tr}) = \boldsymbol{\varphi}_f(\boldsymbol{x}_f^i(\boldsymbol{\mu}_j^{tr}); \boldsymbol{W}_f, \boldsymbol{b}_f) \tag{15}$$

Where $\boldsymbol{W}_f$, $\boldsymbol{b}_f$ are the weight and bias terms of the FFNN and $\boldsymbol{\varphi}_f : \mathbb{R}^{n_\mu+1} \rightarrow \mathbb{R}^q$ is a (non)linear activation function.

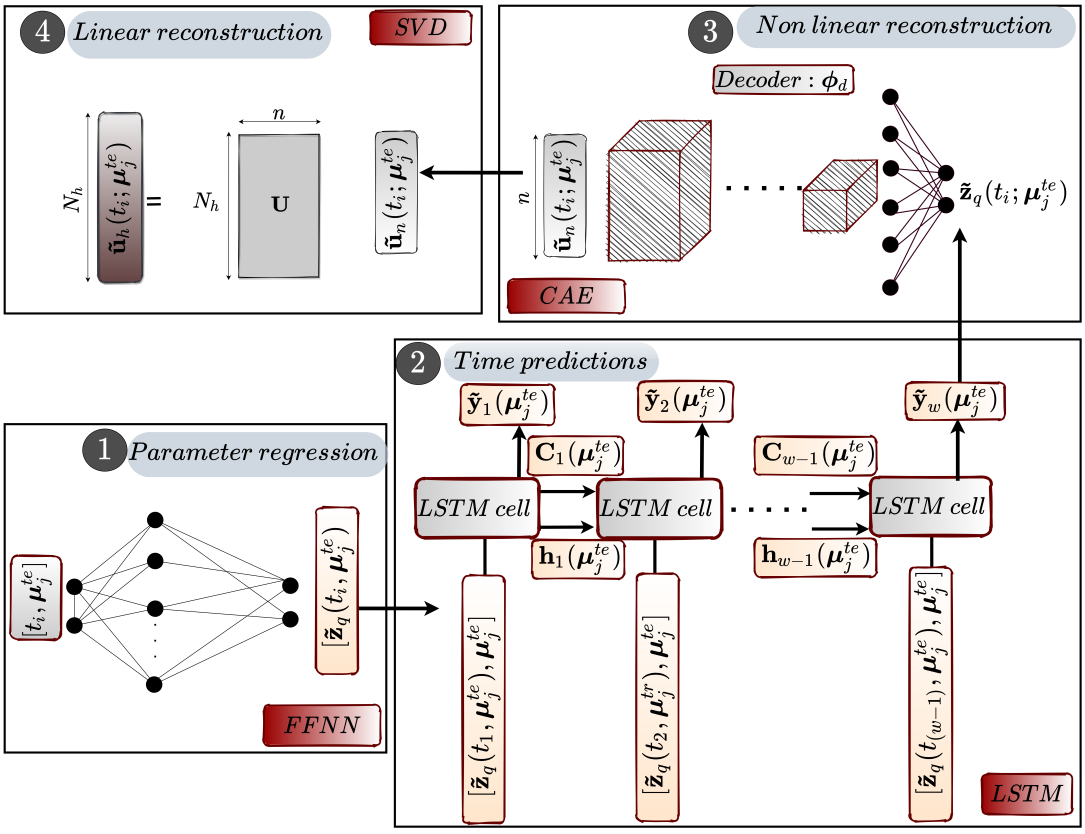

Fig. 2: Online phase (testing) of the *FastSVD-ML-ROM*.

3 Complete scheme of the *FastSVD-ML-ROM* for the UQ

This section summarizes the operation of the *FastSVD-ML-ROM* framework during the offline (training) and online (testing) phase. In the offline phase, the truncated SVD calculates the basis matrix $\boldsymbol{U}$ by giving incrementally partitions of the snapshot matrix $\boldsymbol{S}_h$. Then, the linearly projected data $\boldsymbol{S}_n$ (Eq. 5) is passed to the DL models, which are independently trained. In particular, the CAE is used for the non-linear dimensionallity reduction, the FFNN for the parameter regression and the LSTM for time predictions. The operations of the offline phase are schematically explained in Fig. 1.

In the online phase the input parameter vector $\boldsymbol{\mu}^{te}$ is passed to the trained FFNN, which obtains the first time sequence of the latent variables (Eq. 15) and passes the data to the LSTM network that predicts the evolution of the solution field (Eq. 12). Finally, the decoding part of the CAE (Eq. 8) and the linear reconstruction through the basis matrix $\boldsymbol{U}$ (Eq. 4) are employed to calculate the approximated solutions in a limited amount of time. The operations of the online phase are illustrated in Fig. 2.

Regarding the minimization of the loss function achieved through the backpropagation algorithm, we utilize the adaptive moment estimation optimizer, a version of stochastic gradient descent [20] that employs first and second moment estimates to derive adaptive learning rates for various parameters. All parameters of the DL-models in the frame of *FastSVD-ML-ROM* are tuned through a hyperparameter optimization analysis [21].

Once the *FastSVD-ML-ROM* is trained and tested, it is used to replace the time consuming FEM solvers and estimate the UQ. To compute the effect of the variations introduced in the input parameters (e.g. material properties) to the field of interest, we sample from the same probability distributions used to generate the testing and training parameters, r parameter sets as, $\boldsymbol{\mu}_j^{uq} \in \mathscr{P}$ with $j = 1, \ldots, r$. Therefore, we can predict the output field through the *FastSVD-ML-ROM* and estimate the uncertainty propagation (Fig. 3). In particular, the mean $\boldsymbol{u}_h(t_j, \boldsymbol{\mu}^{uq})_{mean}$ (Eq. 16) and standard deviation $\boldsymbol{u}_h(t_j, \boldsymbol{\mu}^{uq})_{std}$ (Eq. 17) at each time instance t_i with $i =$

$1,\ldots,N_t$, $\forall\ \boldsymbol{\mu}_j^{uq}$ with $j = 1,\ldots,r$ are evaluated.

$$\boldsymbol{u}_h(t_i,\boldsymbol{\mu}^{uq})_{mean} = \frac{1}{r}\sum_{j=1}^{r}\boldsymbol{u}_h(t_i,\boldsymbol{\mu}_j^{uq}) \tag{16}$$

$$\boldsymbol{u}_h(t_i,\boldsymbol{\mu}^{uq})_{std} = \sqrt{\frac{\sum_{j=1}^{r}\left(\boldsymbol{u}_h(t_i,\boldsymbol{\mu}_j^{uq}) - \boldsymbol{u}_h(t_i,\boldsymbol{\mu}^{uq})_{mean}\right)}{r}} \tag{17}$$

Then, the calculation of Eq. 18 follows, for each time instance t_i, to quantify the uncertainties.

$$\boldsymbol{u}_h(t_i,\boldsymbol{\mu}^{uq})_{uq} = \boldsymbol{u}_h(t_i,\boldsymbol{\mu}^{uq})_{mean} \pm \boldsymbol{u}_h(t_i,\boldsymbol{\mu}^{uq})_{std} \tag{18}$$

The overall approach including the implementation of the *FastSVD-ML-ROM* for the estimation of the UQ, followed in this paper is depicted in Fig. 3.

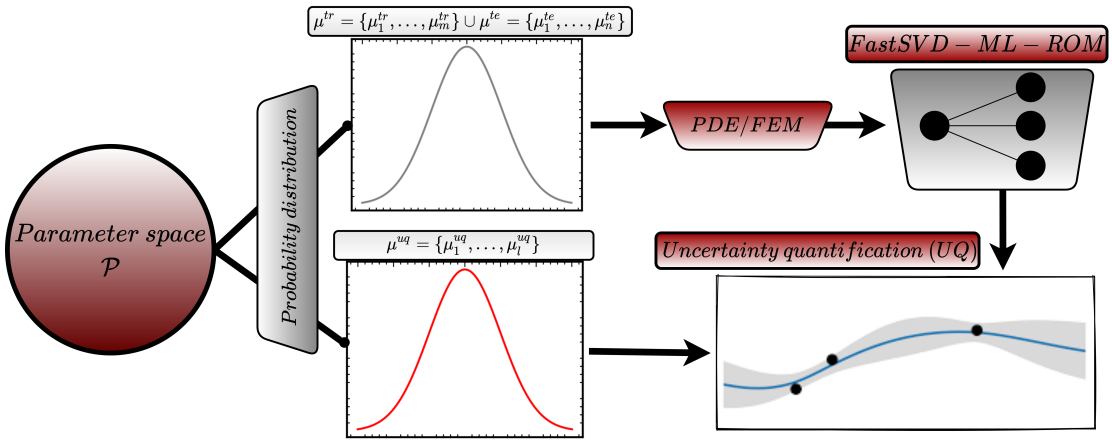

Fig. 3: The integration of the *FastSVD-ML-ROM* framework with the UQ methodology.

We then calculate the indicative damage indices (DIs) proposed in [6]. To achieve that a normalised root meen square deviation (RMSD) DI is defined and computed through the following formula for each parameter value $\boldsymbol{\mu}_j^{uq}$, j =1,...,n and for all t_i, i = 1,..., N_t:

$$DI(t,\boldsymbol{\mu}_j^{uq}) = \frac{\sum_{j=1}^{N_t}\left((\boldsymbol{u}_h(t_i,\boldsymbol{\mu}_j^{uq}) - \boldsymbol{u}_h(t_i,\boldsymbol{\mu}^{uq})_{mean}\right)^2}{\sum_{j=1}^{N_t}\left(\boldsymbol{u}_h(t_i,\boldsymbol{\mu}^{uq})_{mean}\right)^2} \tag{19}$$

where the $\boldsymbol{u}_h(t_j,\boldsymbol{\mu}^{uq})_{mean}$ represents the baseline (healthy) signal.

4 Test case: Guided waves propagation

To validate the proposed ROM approach, the propagation of a lamb wave is simulated in terms of numerical analysis using the FE-method. The numerical model is implemented by utilizing the commercial software SIEMENS Simcenter with NX Nastran [22]. The snapshot matrix is constructed thought the variation of the material properties assuming Gaussian distributions. Experiments are performed on a rectangular aluminum plate under ambient temperature (Fig. 4). The *FastSVD-ML-ROM* framework [18] is employed to obtain the reconstructed full-scale solutions in real-time. The trained NIROM framework, is then used to replace the time consuming and computationally costly, high fidelity solver aiming to address the uncertainty quantification due to the variations introduced in the material properties.

The accuracy of the *FastSVD-ML-ROM* is examined for the testing and training parameters at t_i, with i=1,...,N_t, $\boldsymbol{\mu}_j \in \{\boldsymbol{\mu}_1^{te},\ldots,\boldsymbol{\mu}_n^{te}\} \cup \{\boldsymbol{\mu}_1^{tr},\ldots,\boldsymbol{\mu}_m^{tr}\}$, through the following error indicators,

In particular the truncated SVD update methodology is tested by the l2-norm error ε_{l2} defined as,

$$\varepsilon_{l2} = \frac{\sqrt{\sum_{i=1}^{N_h}(\mathbf{u}_n(t_i;\boldsymbol{\mu}_j^{tr}) - \tilde{\mathbf{u}}_n(t_i;\boldsymbol{\mu}_j^{tr}))^2}}{\sqrt{\sum_{i=1}^{N_h}\mathbf{u}_n(t_i;\boldsymbol{\mu}_j^{tr})^2}}, \tag{20}$$

and the ROM reconstructions are examined using the normalized root mean squared error ε_{nrms} defined as,

$$\varepsilon_{nrms} = \frac{\sqrt{\frac{\sum_{i=1}^{N_h}(\boldsymbol{u}_h(t_i;\boldsymbol{\mu}_j^{te}) - \tilde{\mathbf{u}}_h(t_i;\boldsymbol{\mu}_j^{te}))^2}{N_h}}}{\boldsymbol{u}_h(t_i;\boldsymbol{\mu}_j^{te})^{max} - \boldsymbol{u}_h(t_i;\boldsymbol{\mu}_j^{te})^{min}} \tag{21}$$

4.1 Numerical model set-up

The transient plane stress structural problem is considered to simulate the wake propagation on the plate. The problem is governed by the equation of motion in 3-D linear elasticity as,

$$\frac{\partial^2 u_i}{\partial t^2} = \sum_{k=1}^{3}\left[\frac{E}{2\rho(1+\nu)}\frac{\partial^2 u_i}{\partial x_k \partial x_k} + \frac{E}{2\rho(1-\nu)}\frac{\partial^2 u_k}{\partial x_i \partial x_k}\right],\ i = 1,2,3 \tag{22}$$

where $u_i = u_i(x,y,z)$, i=1,2,3 are the displacement fields in x-, y- and z-axis respectively, E is the young's modulus, ρ is the density and ν is the poisson's ratio, while the body forces are assumed to be zero.

Since the plate is a thin-wall structure, and the ratio of the thickness to the in-plane dimension is high, quadrilateral, isoparametric, plane strain elements are utilised. The finite element model consists of a high-quality, fine mesh grid to enhance numerical convergence and accurate results. In particular, a global mesh size of 0.75 mm is derived through a mesh convergence study, resulting to 82028 nodes. A five-peak tone burst signal with a center frequency 250 kHz is used to excite the plate in the z-axis direction while the left and right edges of the plate are kept fixed as shown in Fig. 4. Regrading the set up of the transient structural dynamics for the wave propagation a total time T = 0.0001 s is defined with a timestep $dt = 1 \times 10^{-7}$ s, resulting in 1000 time solutions. We then chose 200 solutions, with a step equal to 5.

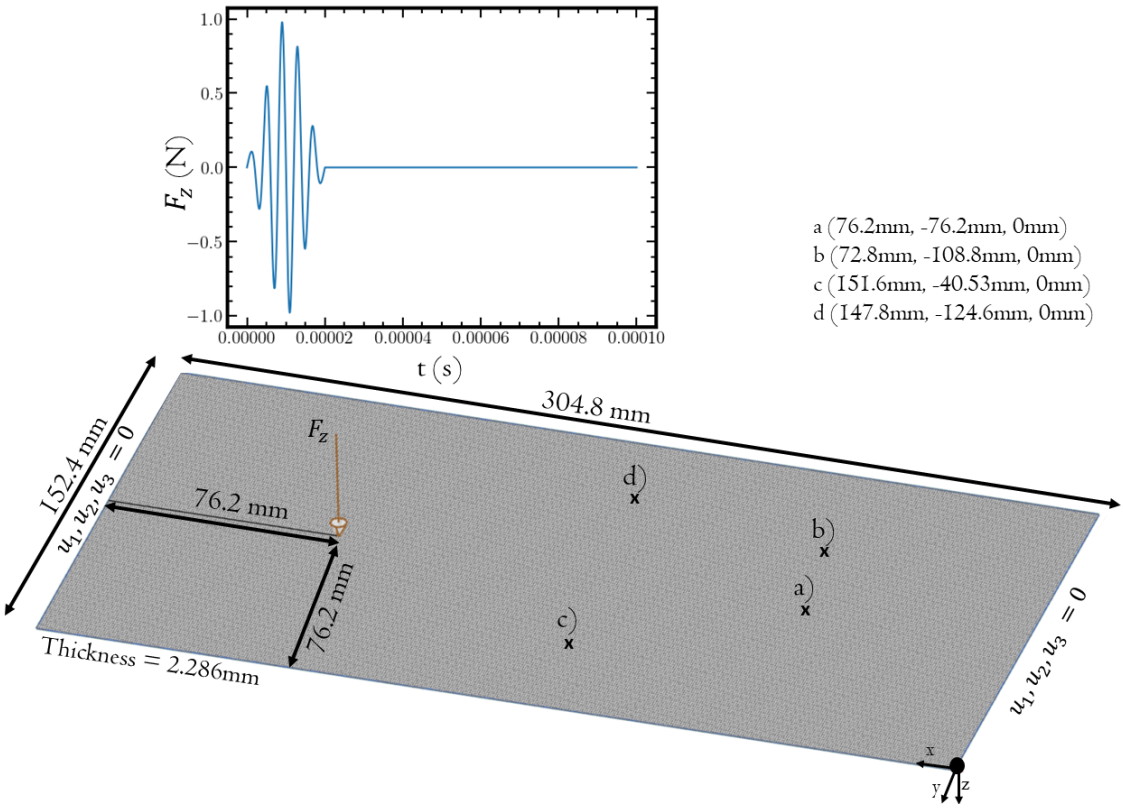

Fig. 4: Geometry and mesh grid of the rectangular plate including the boundary conditions.

To numerically simulate the variations in the material properties on the propagation of lamb waves, we consider three Gaussian probability distributions for the young's modulus (E), the density (ν) and the poisson's ratio (ρ), as well. The mean value of the young's modulus is taken to be 68.9 GPa with a standard deviation of 1.332 GPa. The mean value of the poisson's ratio is 0.33 with a standard deviation of 0.007. As concerns the density, a mean value of 2700 kg/m^3 is selected including a standard deviation of 2.7 kg/m^3.

Each parameter set vector $\boldsymbol{\mu}_j$, used to solve Eq. 22 and generate the HFM solutions is formulated by the triple $\{[E_j, \nu_j, \rho_j]\}$. In particular, the governing equation, is solved for m=24 training parameter sets, while n=2 testing parameters are utilized to test the *FastSVD-ML-ROM*.

4.2 *FastSVD-ML-ROM* results

To construct the low-dimensional basis $\boldsymbol{U}$, we utilize the truncated SVD updated methodology, as presented in [18] during the simulation process. In particular, we define an accuracy $\varepsilon = 4 \times 10^{-4}$ for the truncation of the linear basis vector, resulting to a basis of size 256. To gain a better insight into the performance of the linear projection methodology, we define and present the compression ratio (CPR) as the fraction of the reduced basis dimension to the rank of the original snapshot matrix. The evolution of the CPR as well as the ε_{l2} are presented in Fig. 5. It is shown that the CPR decays rapidly and approximates an exponential behavior, leading to a maximum compression close to 1.56%. Moreover, the evolution of the e_{L2} indicates a growth along with the batches.

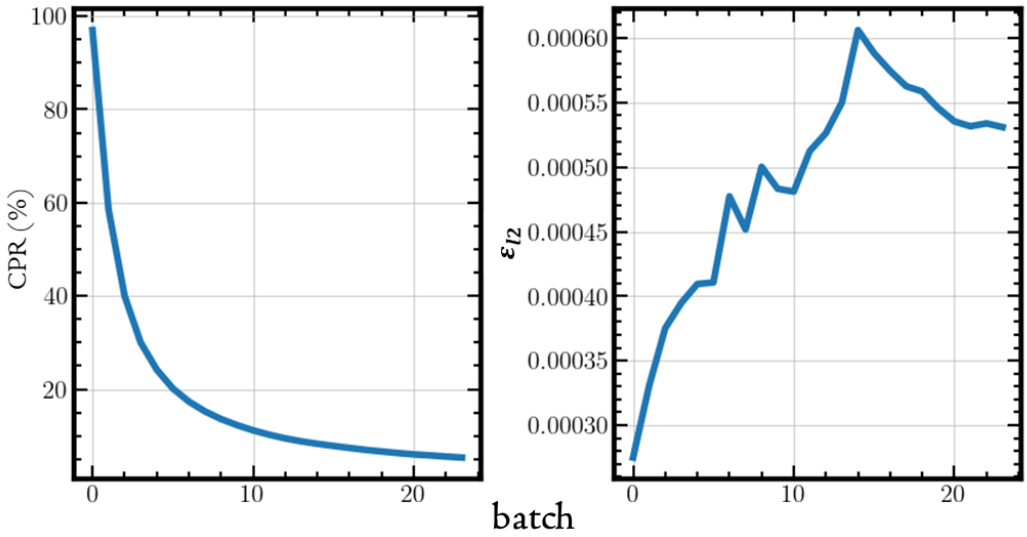

Fig. 5: The implementation of the truncated SVD update. The evolution of the compression ration (CPR) (left) and the ε_{l2} (right) are presented versus the batches using a truncation error $\varepsilon = 4 \times 10^{-4}$.

Table 1: NN-config. and efficiency of the DL-models in terms of accuracy. The GPU computational times of the ROM deployment are presented against the HFM simulation time that requires 3 h for a given parameter set $\boldsymbol{\mu}_i$.

DL-model / **NN-config.**	**CAE**	**FFNN**	**LSTM**
learning rate	0.0001	0.001	0.001
batch	20	5	5
epochs	40000	10000	1000
val. loss	2.7×10^{-6}	6.7×10^{-5}	2.7×10^{-5}
train. loss	2.3×10^{-6}	1.7×10^{-5}	1.7×10^{-4}
training time	55 m	15 m	30 m
testing time	0.05 s	0.01 s	0.043 s
speed up	2.1×10^{5}	1×10^{6}	2.5×10^{5}

Regarding the CAE, a latent space dimension q=4 is pre-defined, resulting in a compression equal to 1.5%. As concerns the dynamics, we set a time window w = 20 for the LSTM network. The detailed description of the configurations used for each DL model are presented in Table 1. The speed-up number is defined as the fraction of the testing time of the ROM and the computational time of the HFM solution.

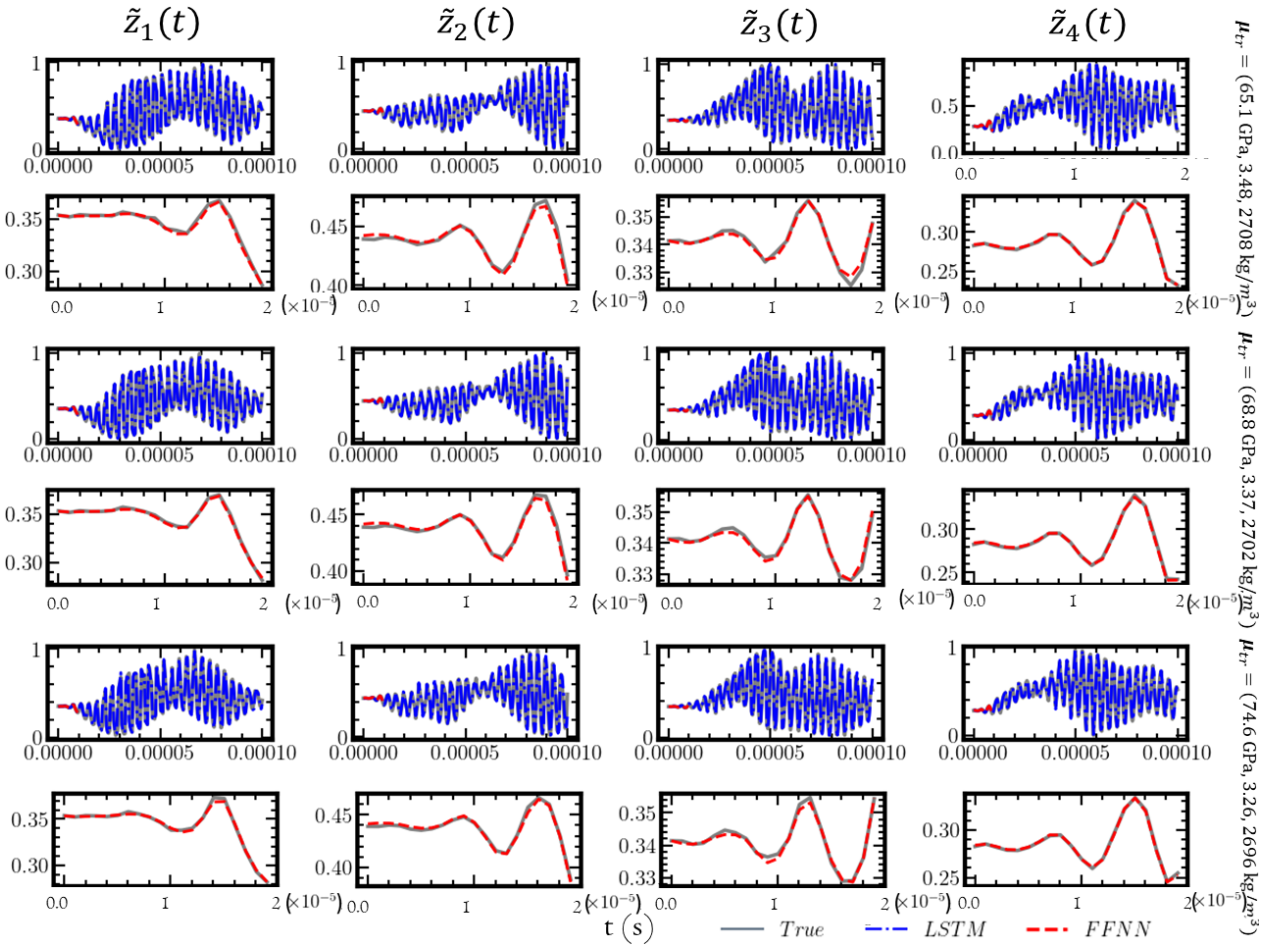

Fig. 6: True latent variables extracted by the trained decoder. LSTM and FFNN predictions (top) until $t = 10^{-4}$ s and FFNN predictions (bottom) until $t = 0.2 \times 10^{-5}$ s for the $\boldsymbol{\mu}^{tr}$ parameters.

To gain a better insight on the ability of the *FastSVD-ML-ROM*, to capture the dynamics, we examine the performance of the LSTM and FFNN (Fig.6). This is achieved by comparing the extracted latent variables from the trained encoder with the FFNN ans LSTM predicted solution versus time. In particular, we compare the result of latent variables evolution $\tilde{\boldsymbol{z}}$ for the testing parameters $\boldsymbol{\mu}^{tr}$=$(65.1\,GPa, 3.48, 2708\,kg/m^3)$, $\boldsymbol{\mu}^{tr}$=$(68.8\,GPa, 3.37, 2702\,kg/m^3)$, and $\boldsymbol{\mu}^{tr}$=$(74.6\,GPa, 3.26, 2696\,kg/m^3)$. It is shown that, the FFNN can accurately predict the first w components of the $\tilde{\boldsymbol{z}}$ temporal evolution, corresponding to the time interval $t = [0, 0.2 \times 10^{-5}]$s, required by the LSTM to predict the entire evolution in time during the online phase, $t = [0, 10^{-4}]$. It is also evident that the LSTM can precisely capture the dynamic behavior until $t = 10^{-4}$ s. The results of the true and predicted results are presented for $\boldsymbol{\mu}^1_{te}$=$(67.2\,GPa, 0.33, 2695\,kg/m^3)$ until $t = 0.000075$ s (Fig. 7, (top)) and $\boldsymbol{\mu}^2_{te}$=$(67.2\,GPa, 0.33, 2695\,kg/m^3)$ at $t = 0.0001$ s (Fig. 7). We note the ability of the *FastSVD-ML-ROM* to accurately predict the acceleration distribution on the plate. In particular, for $\boldsymbol{\mu}^1_{te}$, an error $\varepsilon_{rmse} = 0.06$ at $t = 0.000075$ s is obtained (accuracy in mm) and regarding $\boldsymbol{\mu}^2_{te}$ the $\varepsilon_{rmse} = 0.03$ at $t = 0.0001$ s.

To further access the performance of the *FastSVD-ML-ROM*, in Fig. 8, we present the evolution of the a_z on the point a,b,c, and d (as shown in Fig. 4) for the $\boldsymbol{\mu}^2_{te}$. It is obvious that the *FastSVD-ML-ROM* can accurately monitor the acceleration values in z-axis near and far from the excitation location and effectively model the time-varying (non)linear dynamics, including complex wave interaction phenomena. In Fig. 8, (upper-left), by examining the temporal evolution on node a, the symmetric S_0 and anti-symmetric A_0 modes are evident. In Fig. 8, (upper-left), the S_0 mode is clearly observed while the A_0 seems to be mixed caused by the reflections from the edge. As concerns the temporal evolution of the node c (Fig. 8, bottom-left), it is shown that the S_0 and A_0 modes are observed, followed an another wave packet while the temporal evolution on node d (Fig. 8, bottom-right) presents an overlapping with another wave packet.

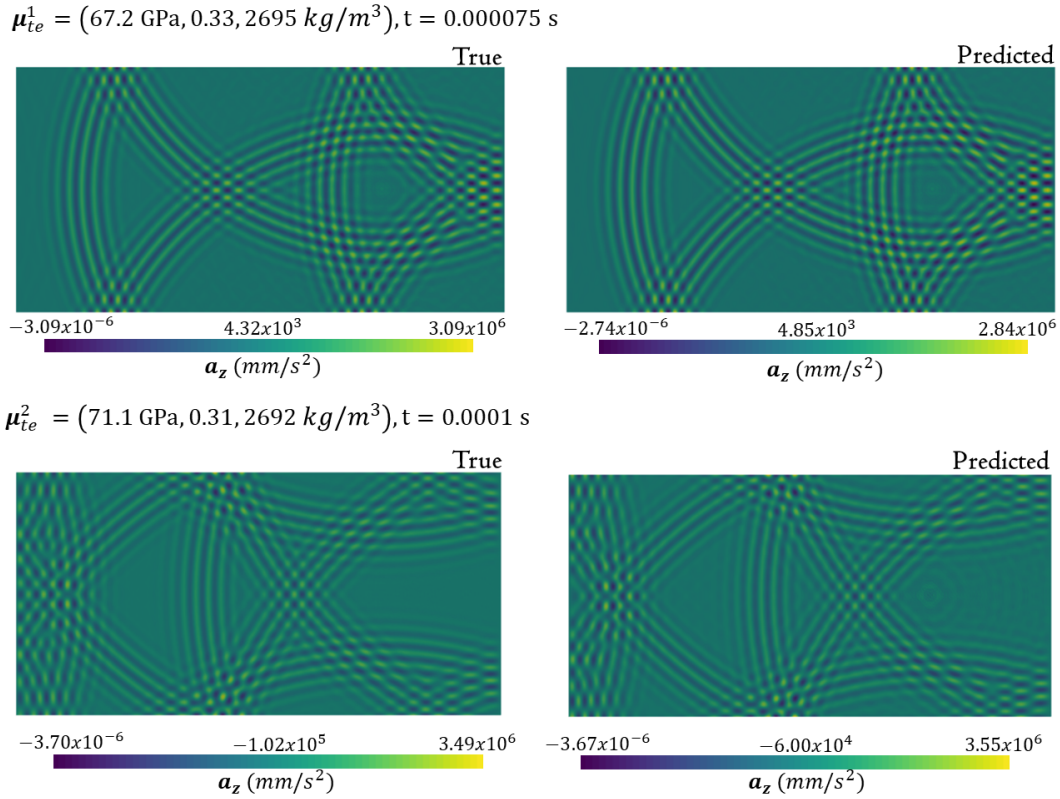

Fig. 7: Comparison between the true and predicted solution at $t = 0.000075s$ (top) for μ_{te}^1=$(67.2\,GPa, 0.33, 2695\,kg/m^3)$ and $t = 0.0001s$ (bottom) for μ_{te}^2=$(71.1\,GPa, 0.31, 2692\,kg/m^3)$.

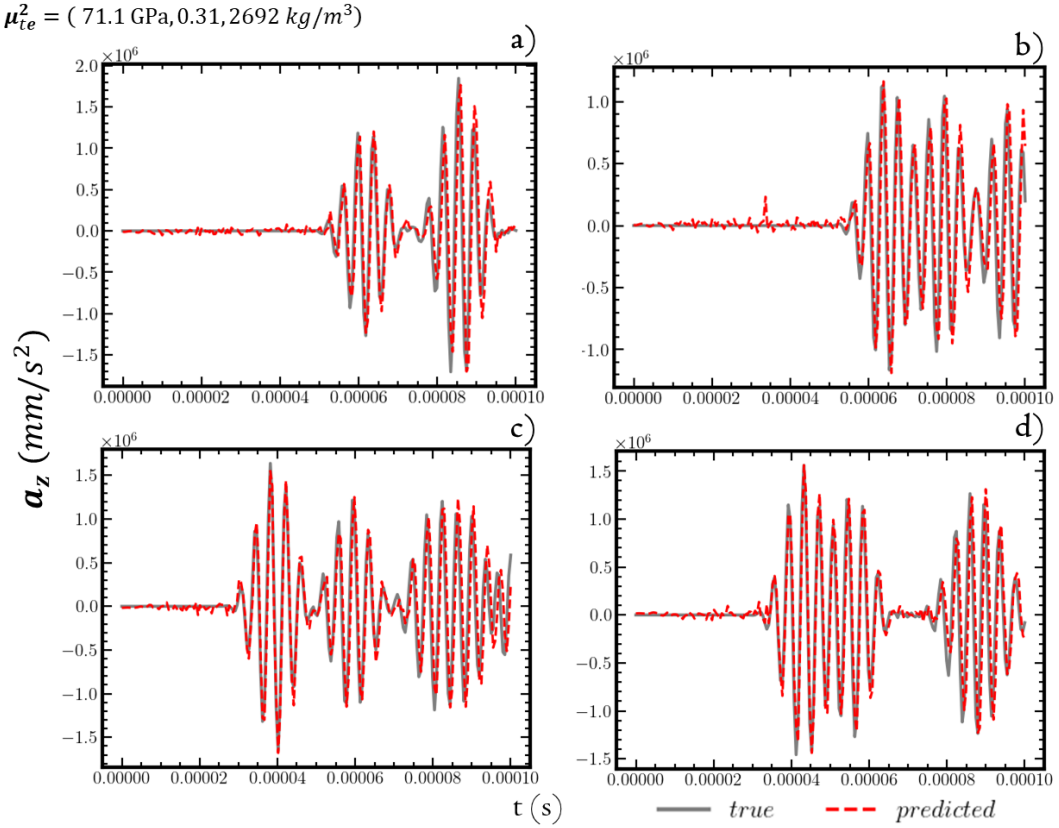

Fig. 8: Monitoring of the true and predicted acceleration solutions on node a,b,c and d for μ_{te}^2.

4.3 Uncertainty quantification implementation

Once the proposed ROM framework, is trained and tested, the uncertainty quantification caused due to the variation introduced into the material properties is examined. To gain a better insight on the effect of the young's modulus E, poisson's ratio ν and density ρ, we sample 1000 ($r = 1000$) parameters $\boldsymbol{\mu}_j^{uq}$ for the UQ investigation. The developed *FastSVD-ML-ROM* is implemented for all the $\boldsymbol{\mu}^{uq}$ at each time-step t_j to compute the acceleration field.

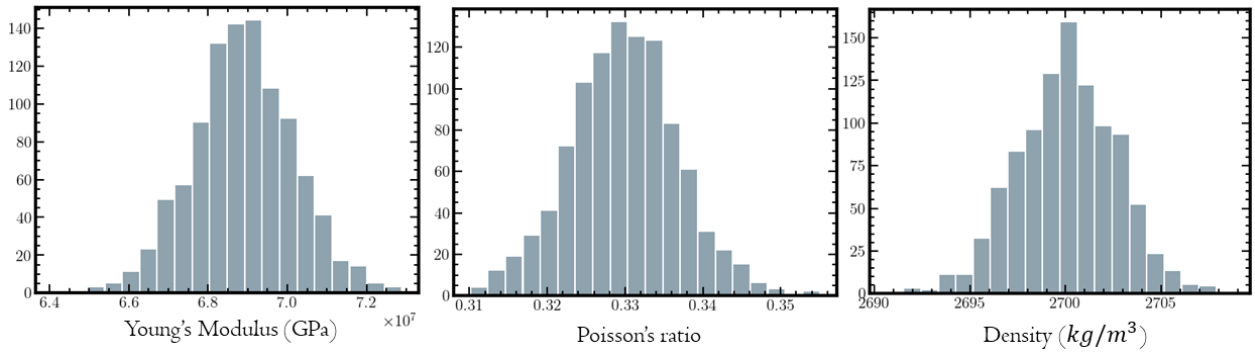

Fig. 9: Histogram of the material properties variations used to generate the uncertainty quantification.

Fig. 10, presents the UQ plots for the 1000 different signals on the nodes a,b,c, and d (Fig. 4), generated by considering the variation in the material properties. The black lines represents the mean value of the time series while the gray area represents the $\pm$ 1 standard deviation of the data. It is observed that the variation in the Young's modulus, the poisson's ration and the density of the aluminum plate, sampled from Gaussian distributions significantly affects the acceleration evolution in z-axis (excitation direction). In particular, it appears that the change in the material properties, mainly affects the amplitude of the signals, with slight variations in their phase.

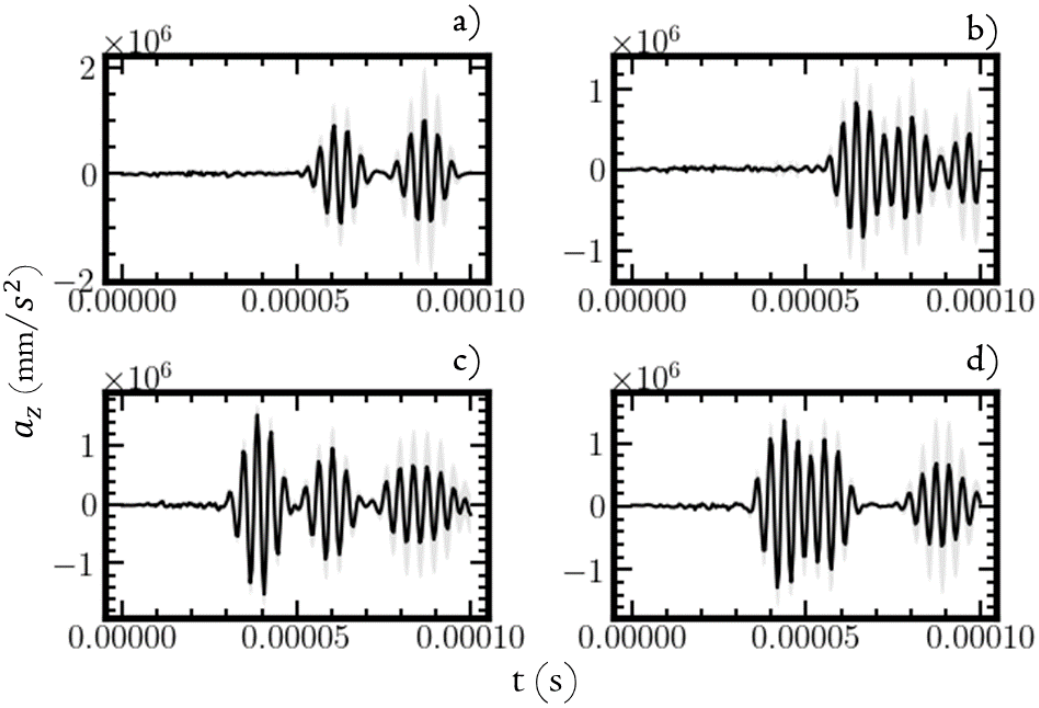

Fig. 10: Mean acceleration (black line) and $\pm$ 1 standard deviation distribution (gray area) for the $\boldsymbol{\mu}^{uq}$ parameters on node a,b,c and d.

We then calculate and present the indicative damage indices (DIs) versus the variations in the material properties on node *c* (Fig. 11). It is shown that the DIs are sensitive to the introduced uncertainties and present higher values as the material properties deviate from their nominal values.

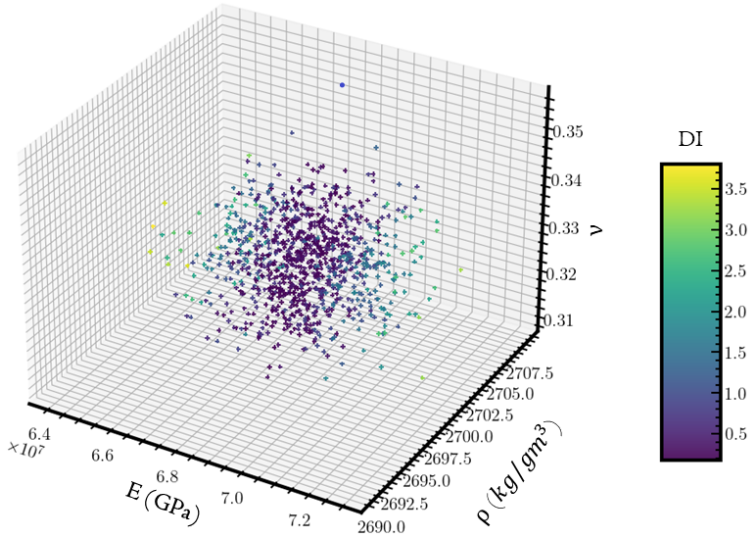

Fig. 11: Damage index variation on node c, caused by the uncertainties introduced in the material properties.

In Fig. 12, the correlations of the DIs distribution with the variations of the material properties are presented. It can be observed that a potential variation occurs between the mean values of E_i, ν_i and ρ_i. The DIs distribution present higher values near the asymptotic areas of each Gaussian distributions corresponding to the uncertainties in the material properties. The lowest values are observed close to the mean value of each distribution.

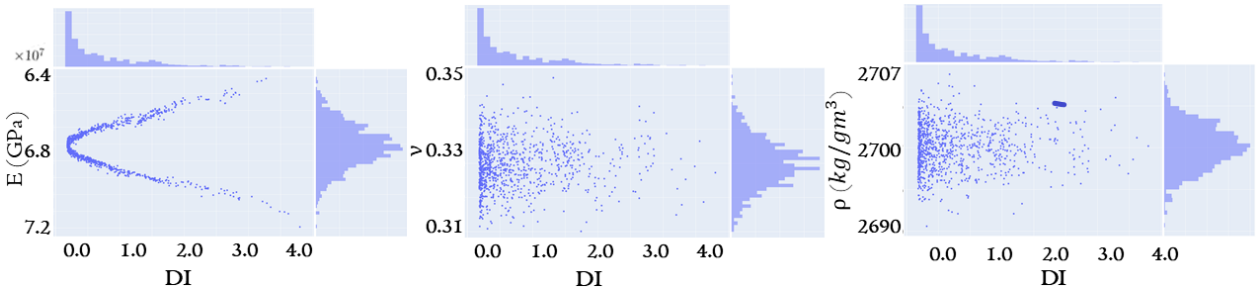

Fig. 12: Correlations of the DI with the young's modulus (left), the poisson's ration (middle) and the density (right) on node c.

5 Conclusions

In this paper, we utilize a NIROM framework, mentioned as *FastSVD-ML-ROM* to replace the computationally costly HFM and compute the UQ of guided waves propagation phenomena by providing solutions in a limited amount of time. The FEM is employed to simulate the structural dynamics problem with great accuracy through the variation of the material properties sampled from Gaussian distributions. Regarding the training and testing of the *FastSVD-ML-ROM*, the comparison among the HFM and ROM solutions, indicates the robustness and efficiency of the applied methodology. As presented, the ROM reconstructions provide accurate results compared with the numerical data, providing a speed up of 5×10^5, in contrast to the NX Nastran solver that require 3h for a high fidelity simulation.

Thanks to a suitable integration of the *FastSVD-ML-ROM* with the forward UQ, the proposed platform is a powerful tool for estimating the effect of the variations introduced in a structural dynamic problem that simulates the propagation of guided waves. The efficiency and flexibility of the developed framework, broadens the horizons for the implementation of the NIROMs in quantifying the uncertainties of the generated data extracted by high fidelity, large-scale computational models.

6 Aknowledgements

The present work is supported by the Karlsruhe Institute of Technology under the project 81227 entitled << DAAD-Development of teaching for the installation and operation of wind turbines computer-aided modeling>> .

References

[1] F. P. Kopsaftopoulos and S. D. Fassois, "Vibration based health monitoring for a lightweight truss structure: experimental assessment of several statistical time series methods," *Mechanical Systems and Signal Processing*, vol. 24, no. 7, pp. 1977–1997, 2010.

[2] P. Ladpli, F. Kopsaftopoulos, and F.-K. Chang, "Estimating state of charge and health of lithium-ion batteries with guided waves using built-in piezoelectric sensors/actuators," *Journal of Power Sources*, vol. 384, pp. 342–354, 2018.

[3] V. Janapati, F. Kopsaftopoulos, F. Li, S. J. Lee, and F.-K. Chang, "Damage detection sensitivity characterization of acousto-ultrasound-based structural health monitoring techniques," *Structural Health Monitoring*, vol. 15, no. 2, pp. 143–161, 2016.

[4] F. J. Gonzalez and M. Balajewicz, "Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems," *arXiv preprint arXiv:1808.01346*, 2018.

[5] T. V. Gortsas, T. Triantafyllidis, S. Chrisopoulos, and D. Polyzos, "Numerical modelling of micro-seismic and infrasound noise radiated by a wind turbine," *Soil Dynamics and Earthquake Engineering*, vol. 99, pp. 108–123, 2017.

[6] S. Ahmed and F. Kopsaftopoulos, "Uncertainty quantification of guided waves propagation for active sensing structural health monitoring," in *Proceedings of the vertical flight society 75th annual forum & technology display, Philadelphia, PA, USA*, pp. 13–16, 2019.

[7] P. Jacquier, A. Abdedou, V. Delmas, and A. Soulaïmani, "Non-intrusive reduced-order modeling using uncertainty-aware deep neural networks and proper orthogonal decomposition: application to flood modeling," *Journal of Computational Physics*, vol. 424, p. 109854, 2021.

[8] S. Pagani and A. Manzoni, "Enabling forward uncertainty quantification and sensitivity analysis in cardiac electrophysiology by reduced order modeling and machine learning," *International Journal for Numerical Methods in Biomedical Engineering*, vol. 37, no. 6, p. e3450, 2021.

[9] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the digital twin: A systematic literature review," *CIRP Journal of Manufacturing Science and Technology*, vol. 29, pp. 36–52, 2020.

[10] A. Arzani and S. T. Dawson, "Data-driven cardiovascular flow modelling: examples and opportunities," *Journal of the Royal Society Interface*, vol. 18, no. 175, p. 20200802, 2021.

[11] A. Rasheed, O. San, and T. Kvamsdal, "Digital twin: Values, challenges and enablers," *arXiv preprint arXiv:1910.01719*, 2019.

[12] Y. Kim, K. Wang, and Y. Choi, "Efficient space–time reduced order model for linear dynamical systems in python using less than 120 lines of code," *Mathematics*, vol. 9, no. 14, p. 1690, 2021.

[13] N. T. Mücke, S. M. Bohté, and C. W. Oosterlee, "Reduced order modeling for parameterized time-dependent pdes using spatially and memory aware deep learning," *Journal of Computational Science*, vol. 53, p. 101408, 2021.

[14] M. Salvador, L. Dede, and A. Manzoni, "Non intrusive reduced order modeling of parametrized pdes by kernel pod and neural networks," *Computers & Mathematics with Applications*, vol. 104, pp. 1–13, 2021.

[15] S. Nikolopoulos, I. Kalogeris, and V. Papadopoulos, "Non-intrusive surrogate modeling for parametrized time-dependent partial differential equations using convolutional autoencoders," *Engineering Applications of Artificial Intelligence*, vol. 109, p. 104652, 2022.

[16] S. Fresca and A. Manzoni, "Pod-dl-rom: enhancing deep learning-based reduced order models for nonlinear parametrized pdes by proper orthogonal decomposition," *Computer Methods in Applied Mechanics and Engineering*, vol. 388, p. 114181, 2022.

[17] T. Nakamura, K. Fukami, K. Hasegawa, Y. Nabae, and K. Fukagata, "Convolutional neural network and long short-term memory based reduced order surrogate for minimal turbulent channel flow," *Physics of Fluids*, vol. 33, no. 2, p. 025116, 2021.

[18] G. I. Drakoulas, T. V. Gortsas, G. C. Bourantas, V. N. Burganos, and D. Polyzos, "*FastSVD-ML-ROM*: A reduced-order modeling framework based on machine learning for real-time applications," *arXiv preprint arXiv:2207.11842*, 2022.

[19] S. Yan, "Understanding lstm networks," *Online). Accessed on August*, vol. 11, 2015.

[20] N. Ketkar, "Stochastic gradient descent," in *Deep learning with Python*, pp. 113–132, Springer, 2017.

[21] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, 2020.

[22] N. N. U. Guide, "Siemens plm software," *United States*, 2014.