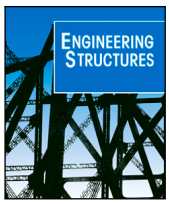
Engineering Structures 270 (2022) 114811

Contents lists available at ScienceDirect

Engineering Structures

journal homepage: www.elsevier.com/locate/engstruct



Assessment of DeepONet for time dependent reliability analysis of dynamical systems subjected to stochastic loading

Shailesh Garg [a], Harshit Gupta [b], Souvik Chakraborty [a,c,*]

[a] Department of Applied Mechanics, Indian Institute of Technology Delhi, Hauz Khas 110 016, India
[b] Department of Mechanical Engineering, Indian Institute of Technology Delhi, Hauz Khas 110 016, India
[c] School of Artificial Intelligence (ScAI), Indian Institute of Technology Delhi, Hauz Khas 110 016, India

ARTICLE INFO

Dataset link: https://github.com/csccm-iitd/DeepONet-reliability

Keywords:
DeepONet
Neural Networks
Stochastic ODE
Nonlinear systems

ABSTRACT

Time dependent reliability analysis and uncertainty quantification of structural system subjected to stochastic forcing function is a challenging endeavour as it necessitates considerable computational time. We investigate the efficacy of recently proposed DeepONet in solving time dependent reliability analysis and uncertainty quantification of systems subjected to stochastic loading. Unlike conventional machine learning and deep learning algorithms, DeepONet is an operator network and learns a function to function mapping and hence, is ideally suited to propagate the uncertainty from the stochastic forcing function to the output responses. We use DeepONet to build a surrogate model for the dynamical system under consideration. Multiple case studies, involving both toy and benchmark problems, have been conducted to examine the efficacy of DeepONet in time dependent reliability analysis and uncertainty quantification of linear and nonlinear dynamical systems. Comparisons have also been drawn with Recurrent Neural Network results and with results obtained from Proper Orthogonal Decomposition based Gaussian process. The results obtained indicate that the DeepONet architecture is accurate as well as efficient. Moreover, DeepONet posses zero shot learning capabilities and hence, a trained model easily generalizes to unseen and new environment with no further training.

1. Introduction

While designing any engineering structure, performing appropriate tests to check the validity of the designs under various circumstances is of utmost importance. For a few cases where the forces acting on the system are predictable, analysis can be performed on a fixed set of samples to verify the design of the structure. However, in a majority of cases, the forces acting on the system are random in nature, and it is difficult for the designer to explicitly designate the design as pass or fail. For such cases, quantifying the uncertainties in the system, assessing its influence on system responses, and performing reliability analysis [1,2] becomes utmost importance.

Reliability analysis at its core entails computing failure probability and several methods [3] exists which can compute the required probability albeit with varying accuracy. Among various methods, simulation based methods are widely accepted to be most accurate. Simulation based methods involve computing large number of samples corresponding to the uncertain parameters in the system and to perform such a comprehensive analysis, both time and computational power are required, a shortage of which can lead to inadequate and unreliable design. To perform time dependent reliability analysis of dynamical systems, in the face of uncertain forcing conditions, a large number of input force realizations are generated, and a corresponding displacement ensemble is computed. We essentially test the system against a range of input forces to calculate its failure probability. Traditionally, system states for individual realizations of input force have been generated using integration schemes [4,5] like Runge–Kutta Method [6] and Newmark Beta method [7,8]. Techniques like Monte Carlo simulation [9], importance sampling [10,11], Latin Hypercube Sampling [12] are a few of the popular simulation based techniques. To reduce the associated computational cost, subset simulation [13,14] has been used in the past, but the same is not suited for first passage failure time analysis. These schemes, find their use-case limited due to the associated computational cost.

In order to reduce the computational requirements, one may choose to decrease the dimensionality of the data using techniques like Principal Component Analysis [15], Independent Component Analysis [16] or other such techniques [17], but however good the technique may be there is bound to be some information loss because of model order reduction [18]. To bypass these problems, one may use machine learning [19,20] in conjunction with the traditional integration techniques.

* Corresponding author at: Department of Applied Mechanics, Indian Institute of Technology Delhi, Hauz Khas 110 016, India.
E-mail addresses: shaileshgarg96@gmail.com (S. Garg), harshit.gupta.iitd23@gmail.com (H. Gupta), souvik@am.iitd.ac.in (S. Chakraborty).

https://doi.org/10.1016/j.engstruct.2022.114811
Received 18 January 2022; Received in revised form 7 July 2022; Accepted 7 August 2022
Available online 2 September 2022

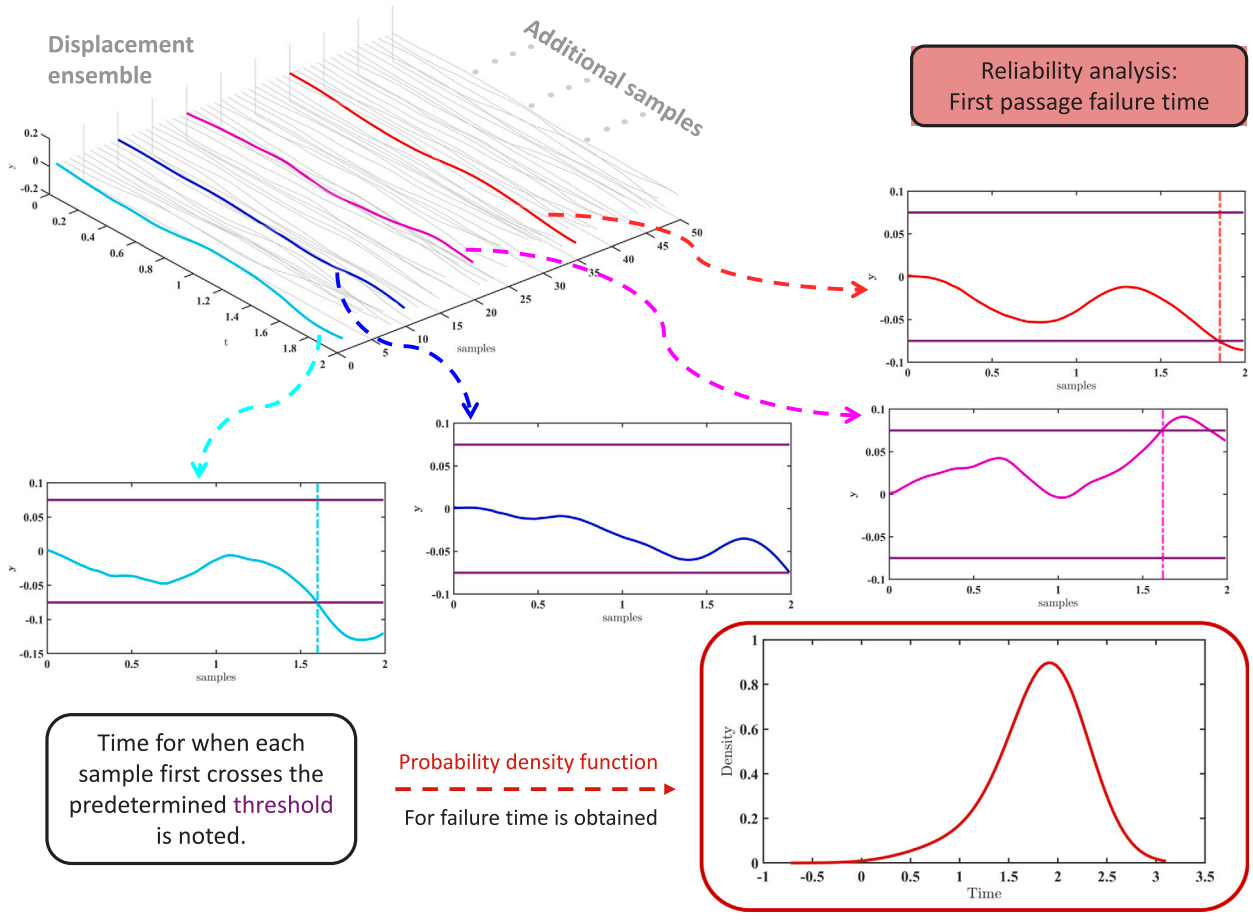


Fig. 1. Pictorial representation of steps involved in reliability analysis. Specifically *first passage failure time* is computed. A displacement ensemble is generated using realizations of random input force. Each sample from the resulting ensemble is tested against a predetermined threshold, and the first time the threshold is crossed is noted for each sample. If the threshold is not crossed, the final time for that sample is recorded. The probability density function for the resulting time vector is obtained, and thus the probability of failure of the structure at any given time can be obtained.

Machine learning is a budding field of research and has found its use across several industries [21–24]. Machine learning techniques, especially neural networks [25,26] (NN) have the ability to learn from a small training data set and then apply that same knowledge to a vast data set with very little computational cost associated with it. Owing to this, several NN architectures have been developed with an aim to solve a set of differential equations or to learn nonlinear operators. Popular examples include Physics Informed NN [27], Fourier NN [28] and DeepONet [29].

In this paper, we propose using DeepONet, a neural network architecture, to prepare a surrogate model which can then be used to generate system states for multiple test samples with reduced computational cost. DeepONet was proposed by Lu et al. in their paper [29], and it aims at learning the nonlinear operator based on a limited training data set. It is a data-driven architecture based on the universal approximation theorem [30] and can reliably solve ordinary differential equations and stochastic differential equations [31] (SDE). Results generated in this paper also indicate that given sufficient training data, DeepONet model demonstrates Zero-Shot Learning (ZSL) [32] capabilities for new inputs; this allows generalization of the proposed architecture.

The proposed surrogate model, while applicable for any dynamical system, is especially useful in cases where simulating high fidelity data is complex, and only a small set of the same can be generated. Results produced show that by only simulating a few hundred samples, thousands of samples can be inferred using the surrogate model with reasonable accuracy. It also gives the user ability to produce only relevant results i.e. if lets say displacements only at Nth DOF are required, traditional integration schemes require the displacements and velocities to be computed at all DOFs but using the trained surrogate model, user will be able to generate only the required data thus saving expensive server space. Additionally, for the cases where the simulation algorithm converges only for *fine* time steps, the proposed surrogate model can be trained at *coarser* time steps while retaining the integrity of the predictions, thus saving both computational cost and time. While other machine learning based surrogate algorithms for uncertainty quantification or analysis of dynamical systems are available in the literature which adopt Gaussian process or Kriging [33–35], NNs [36] and other such techniques [37–45], no work investigating the applicability of DeepONet for time-dependent reliability analysis and uncertainty quantification under forcing function uncertainty exists in the open literature, and owing to the benefits of DeepONet listed above, it is worthwhile to investigate its applicability on the same.

The final results produced in case studies, using the proposed DeepONet surrogate model are compared against those obtained using Recurrent Neural Networks [46,47] (RNN). RNNs are a branch neural network architectures which is suited for sequence to sequence mapping and hence are suitable to train time series data similar to that observed for dynamical systems. Results have also been generated using Proper Orthogonal Decomposition [48,49] based Gaussian process (POD-GP) and the same have been compared against DeepONet results.

The remaining paper is arranged as follows, Section 2 discusses the problem statement of the paper while Section 3 introduces the DeepONet architecture being implemented. Section 4 showcases the various case studies carried out to test the proposed surrogate model and Section 5 implements the proposed algorithm on an engineering

examples. Finally, Section 6 draws the appropriate conclusion of the paper.

2. Problem statement

Consider an N-DOF stochastic dynamical system with governing equation as follows:

$$\mathbf{M}\ddot{\boldsymbol{X}} + \mathbf{C}\dot{\boldsymbol{X}} + \mathbf{K}\boldsymbol{X} + \boldsymbol{N}(\boldsymbol{X}, \dot{\boldsymbol{X}}, \boldsymbol{\theta}_N) = \boldsymbol{F}, \tag{1}$$

where $\mathbf{M}$, $\mathbf{C}$ and $\mathbf{K}$ are the mass, damping and stiffness matrices respectively. $\boldsymbol{N}(\cdot)$ is the restoring force which is a nonlinear function of system states $\boldsymbol{X}$ and parameter $\boldsymbol{\theta}_N$. $\boldsymbol{F} \sim \mathbb{P}(\cdot)$ is the input force vector drawn from the probability distribution $\mathbb{P}(\cdot)$. This uncertainty in the input force of Eq. (1) lends it its stochastic nature, thus necessitating reliability analysis.

Now, reliability analysis can be performed using various different techniques [3], choice of which depends on the need of the specific application. The first step, however is usually to draw realizations of input force from the respective probability distribution and solve Eq. (1) to generate a displacement ensemble. At which point, the methodologies diverge depending on the quantity being measured. In this paper, we will be computing First Passage Failure Time [50] (FPFT), which gives a Probability Density Function (PDF) for the failure time of the structure. Steps involved include taking realizations of displacement ensemble one by one and comparing them with the predetermined threshold. The time step at which each sample crosses the threshold first is noted, and for samples that never cross the time step the final time is noted. PDFs for the resulting failure time vector are computed, and further design related decisions can then be made. AThe thresholds in a practical scenario are computed based on the design codes being followed and the type of structure being designed. A schematic of the steps involved in FPFT reliability analysis is shown in Fig. 1. As discussed earlier, the goal of this paper is to prepare a surrogate model of the Eq. (1), which can expeditiously generate results for a wide range of input forces. This will enable us to perform reliability analysis in a short amount of time, giving a thorough picture of the expected system behaviour under various loading conditions. To prepare the aforementioned surrogate model, DeepONet architecture for learning nonlinear operators has been implemented. The neural network architecture is discussed in detail in the following section.

3. DeepONet

DeepONet is a neural network architecture developed to learn nonlinear continuous operators such as integrals, ODEs and stochastic ODEs. It is based on the universal approximation theorem, which states that even a single layer neural network is sufficient to learn a continuous nonlinear operator. In order to reduce the generalization error, DeepONet uses a branch and trunk architecture. The branch network is trained to encode the input function of the nonlinear operator while the trunk network encodes the domain of the input function. The two layers are then merged, and an output is generated. Schematic for DeepONet architecture is shown in Fig. 2, mathematical representation of the same is as follows:

$$G(f)(y) \simeq w\left\langle \alpha_b\left(b_b + \sum_i x_i^b w_i^b\right), \alpha_t\left(b_t + \sum_i x_i^t w_i^t\right)\right\rangle + b, \tag{2}$$

where $G(f)(y)$ is the target nonlinear operator with input f and output y. The quantities $\alpha_b\left(b_b + \sum_i x_i^b w_i^b\right)$ and $\alpha_t\left(b_t + \sum_i x_i^t w_i^t\right)$ are the outputs of branch and trunk network respectively. α_b and α_t in Eq. (2) represent the activation functions for the branch and trunk networks and, b_b and b_t are their respective bias. x^b and x^t are neurons corresponding to last layer of branch and trunk network having weights w^b and w^t respectively. $\langle\cdot,\cdot\rangle$ represents the dot product. w and b are the weight and bias of the final output node, placed after the dot product. Branch and trunk networks can also be configured in such a way that individually they are stacked neural networks. In this paper, we use unstacked DeepONet architecture which takes forcing function as input for branch network and random time step as input for the trunk network. Subsequently the output is the displacement corresponding to the input time step and the forcing function. The training data set for the used DeepONet architecture looks as follows:

$$\begin{bmatrix} f_1^1 & f_2^1 & f_3^1 & . & . & . \\ f_1^2 & f_2^2 & f_3^2 & . & . & . \\ f_1^3 & f_2^3 & f_3^3 & . & . & . \\ . & & & & & \\ . & & & & & \\ . & & & & & \end{bmatrix}, \begin{bmatrix} t_i \\ t_j \\ t_k \\ . \\ . \\ . \end{bmatrix}, \begin{bmatrix} y_i^1 \\ y_j^2 \\ y_k^3 \\ . \\ . \\ . \end{bmatrix}, \tag{3}$$

输出是相应的位移

where $t_i \sim \mathbb{U}(0, t_{end})$ and y_i is the corresponding displacement. Characters in the superscript of f and y represent the sample numbers while in the subscript represent the time step. The loss function for the network is taken as the mean squared error ϵ function which is defined as follows:

网络的损失定义为均方误差

$$\epsilon = \frac{1}{N_p}\sum_{i=1}^{N_p}(y_a - y_p)^2, \tag{4}$$

where y_a is the target output of $G(f)(y)$ and y_p is the output obtained from the neural network. N_p is the number of points and is equivalent to batch size in NN training. Algorithm 1 includes the steps involved in training the surrogate model. We have implemented the DeepONet architecture using Google's Tensorflow [51] machine learning package. The process flow for the proposed algorithm using DeepONet as the surrogate model is shown in Fig. 3. A detailed schematic describing the process followed is shown in Fig. 4.

Algorithm 1: DeepONet Algorithm

```
Prepare the input data set;                          ▷ Refer Eq. (3)
for i = 1: no. of iterations do
    for j = 1: no. of training data batches do
        Input one batch of training force into branch net.
        Input corresponding batch of random time steps into
          trunk net.
        Merge the outputs from branch and trunk net; ▷ Refer
          Eq. (2)
        Compute training error;                      ▷ Refer Eq. (4)
        Update weights and biases based on back-propagation
          with an aim to reduce the training error.
    end
end
```

Remark. One of the distinct features of DeepONet architecture is zero-shot learning (ZSL). In machine learning, ZSL refers to the ability of the trained model to make predictions for which no training data is available. In the context of this paper, ZSL refers to the ability of the DeepONet architecture to make predictions for inputs which have statistical properties different to those inputs, which were used while training the network.

4. Numerical illustration

This section covers two different case studies exploring both single Degree of Freedom (DOF) and multi DOF system with varying nonlinearity. Case-I covers a single DOF Bouc Wen system [52,53] while Case-II deals with a 5-DOF system installed with a duffing oscillator at first DOF. Training data for the DeepONet in the following case studies is sampled at a sampling frequency of 100 Hz for 2 s. Input force applied in Case I(a) and II is generated as follows:

$$f = \sum_{i=1}^{n_s} a_{s_i} sin(f_{s_i} t) + \sum_{i=1}^{n_c} a_{c_i} cos(f_{c_i} t), \tag{5}$$

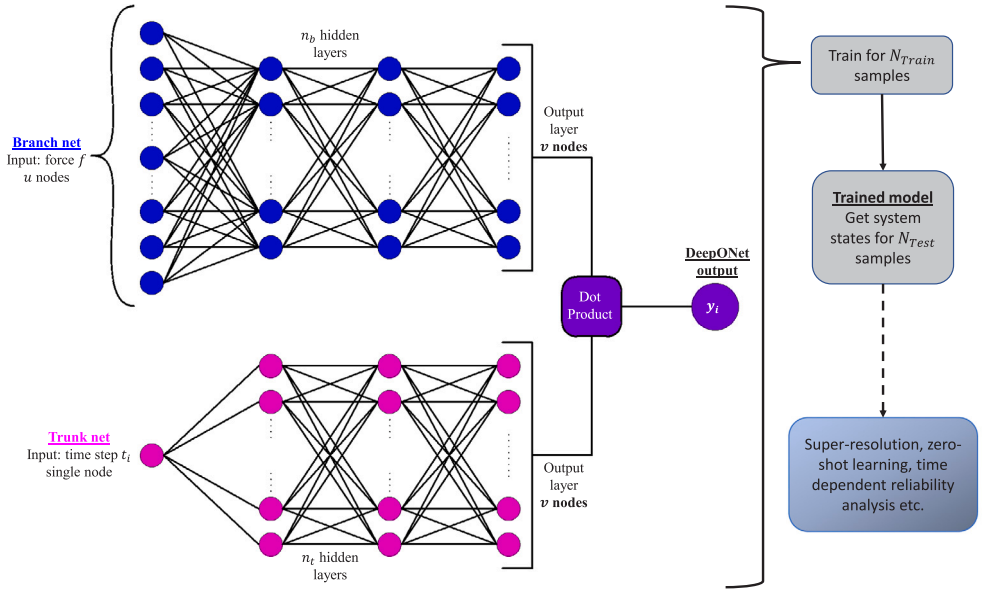


Fig. 2. Flow chart showcasing the DeepONet architecture and how it has been used in the paper. DeepONet's architecture is divided into two main parts namely branch net and trunk net. The branch net takes force f as input while the trunk net takes a random time step t_i as input. Respective inputs are then passed through branch net and trunk net, both having *same output size*. The corresponding outputs obtained are then combined and the final output in the form of displacement y_i corresponding to input time step t_i is obtained. Note, a normalization layer has been added in both branch and trunk net after the initial input layers.

where a_s and a_c are the amplitudes for sine and cosine wave respectively. f_s and f_c are frequencies for sine and cosine wave respectively. a_s, a_c, f_s and f_c are uniform random variables where $f_s, f_c \sim \mathbb{U}(0, 10)$ for all the case studies. $a_s, a_c \sim \mathbb{U}(-50, 50)$ for the first case study, $a_s, a_c \sim \mathbb{U}(-10, 10)$ for the second case study and $a_s, a_c \sim \mathbb{U}(-1, 1)$ for the engineering example covered in the following section. n_s and n_c in Eq. (5) are the number of sine and cosine terms which together $n = n_s + n_c$ are referred to as Fourier terms (FT). Note, $n_s = n_c$ for even FT and $n_s = n_c + 1$ for odd number of FT. PDFs for the input force at a given time step, corresponding to different numbers of FT, is shown in Fig. 5. Forcing function for Case I(b) is drawn from a Gaussian Process having squared exponential as its kernel function. Parameter details for the kernel and mean function are discussed in detail in the case study itself.

DeepONet architecture selected for the following case studies is as follows:

1. **Branch net:** 100 → Batch Normalization → 40 → 40.
2. **Trunk net:** 1 → Batch Normalization → 40 → 40.
3. **Output:** Dot product → 1.

All layers except the final output layer with single node have ReLu activation function [54]. Glorot normal function [55] is used for initializing the weights. Unlike conventional ML algorithms, DeepONet generates function to function mapping (instead of variables to variables); this is why input for branch net is a realization of input force, which has been discretized into 100 points, thus necessitating 100 nodes in the input layer of branch net. The input size will change if forcing function is provided at finer or coarser intervals. The choice of number of hidden layer nodes is subjective and will change according to the needs of the application. For the following case studies however, nodes in hidden layers have been kept same.

The results produced using DeepONet surrogate model are compared against those obtained using RNN and POD-GP. Unless specified otherwise, the RNN layer accepts a single input and extracts 50 features which are then combined in a single Dense output node. ADAM optimizer with learning rate of 10^{-3} and weight decay of 10^{-4} was used to train the RNN; and a total of 5000 epochs were run with a batch size of 10. As for the POD-GP analysis, approximately 95% information was retained [56–58] during dimension reduction and squared exponential kernel was considered for the GP model.

Table 1
System parameters for Case-I: SDOF Bouc Wen Oscillator.

Mass (kg)	Stiffness (N/m)	Damping (N s/m)	Nonlinear parameters
$m = 6800$	$k = 232\,000$	$c = 3750$	$Q_y = 0.05$ mg, $k_r = \frac{1}{6}, \alpha = 1,$ $\beta = 0.5, \gamma = 0.5, D_y = 0.0013, \eta = 2$

4.1. Case-I: SDOF Bouc Wen System

For the first case study, the applicability of the surrogate model is tested for an SDOF Bouc Wen system. Governing equations for the system under consideration are as follows:

$$\begin{aligned} &m\ddot{x} + c\dot{x} + kx + (1 - k_r)Q_y z = f \\ &\dot{z} = \frac{1}{D_y}(\alpha\dot{x} - \gamma z|\dot{x}\parallel z|^{\eta-1} - \beta\dot{x}|z|^{\eta}) \end{aligned} \tag{6}$$

where m, c and k are the mass, stiffness and damping matrices. Q_y, k_r, α, β, γ, D_y and η are the parameters of the Bouc Wen oscillator. Readers interested to read more about Bouc Wen oscillator may refer [52,53]. System parameters are selected from [59] and are given in Table 1.

4.1.1. Case-I (a)

Input forces used in training the DeepONet model for part *a* of Case-I have been generated using Eq. (5) with 20 FT. Testing and training data have been simulated based on Eq. (6) and initial conditions have been taken as $x_0 = 0.01$ m, $\dot{x}_0 = 0.05$ m/s and $z_0 = 0.001$. Comprehensive testing has been carried out to test the surrogate model against a variety of setups and develop an intuition as to what works and why.

The Root Mean square error (RMSE) values in Table 2 showcase the performance of surrogate model when trained for varying number of training samples. Unique Input Realizations (UIR) used to form the training dataset vary from 25 to 150 with time steps per realization or Points Per Realization (PPR) fixed at 100. For n input realizations, discretized at d locations, training dataset generated taking m PPR can

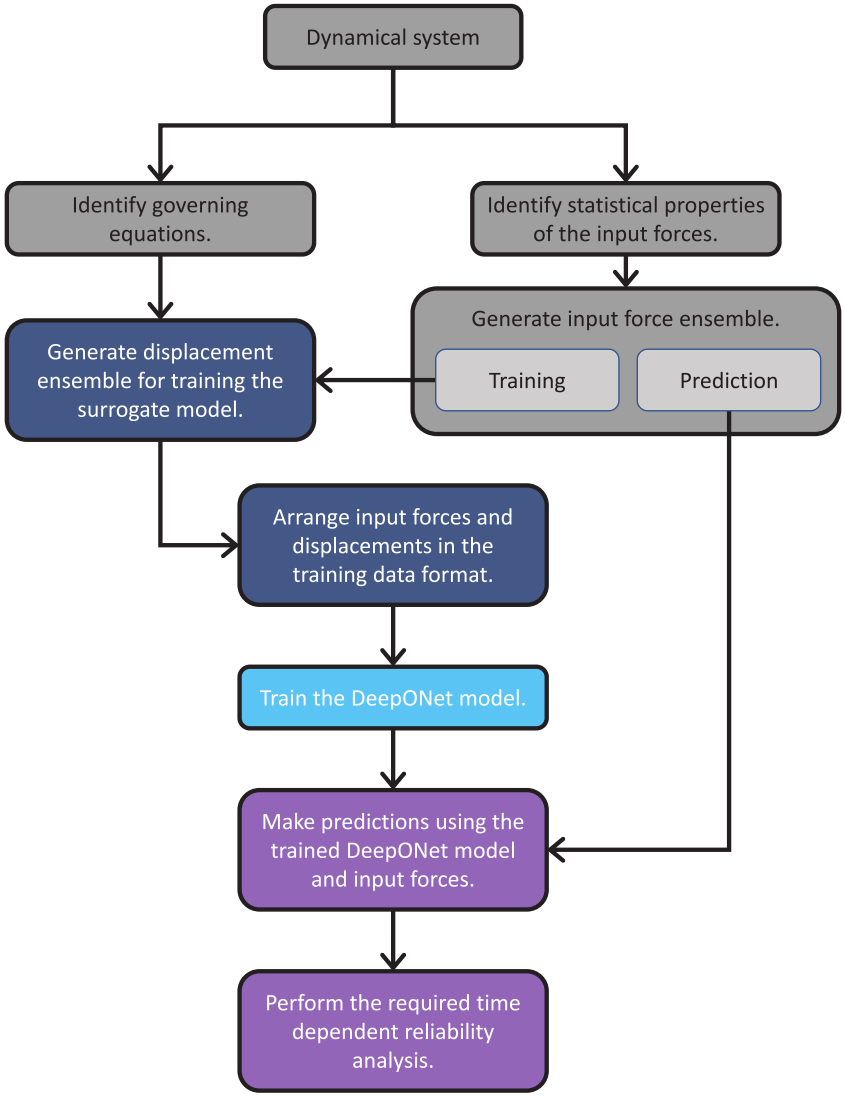


Fig. 3. Flowchart for the time dependent reliability analysis using DeepONet as the surrogate model.

be visualized as follows:

$$\begin{bmatrix} f_1^{(1)} & f_2^{(1)} \cdots & f_{d-1}^{(1)} & f_d^{(1)} \\ f_1^{(1)} & f_2^{(1)} \cdots & f_{d-1}^{(1)} & f_d^{(1)} \\ \vdots & & & \\ f_1^{(1)} & f_2^{(1)} \cdots & f_{d-1}^{(1)} & f_d^{(1)} \\ f_1^{(2)} & f_2^{(2)} \cdots & f_{d-1}^{(2)} & f_d^{(2)} \\ f_1^{(2)} & f_2^{(2)} \cdots & f_{d-1}^{(2)} & f_d^{(2)} \\ \vdots & & & \\ f_1^{(2)} & f_2^{(2)} \cdots & f_{d-1}^{(2)} & f_d^{(2)} \\ \vdots & & & \\ f_1^{(n)} & f_2^{(n)} \cdots & f_{d-1}^{(n)} & f_d^{(n)} \\ \vdots & & & \\ f_1^{(n)} & f_2^{(n)} \cdots & f_{d-1}^{(n)} & f_d^{(n)} \\ f_1^{(n)} & f_2^{(n)} \cdots & f_{d-1}^{(n)} & f_d^{(n)} \end{bmatrix}_{nm\times 100}, \quad \begin{bmatrix} t_1^{(1)} \\ t_2^{(1)} \\ \vdots \\ t_m^{(1)} \\ t_1^{(2)} \\ t_2^{(2)} \\ \vdots \\ t_m^{(2)} \\ \vdots \\ t_{(1)}^{(n)} \\ \vdots \\ t_{(m-1)}^{(n)} \\ t_{(m)}^{(n)} \end{bmatrix}_{nm\times 1}, \quad \begin{bmatrix} y_1^{(1)} \\ y_2^{(1)} \\ \vdots \\ y_m^{(1)} \\ y_1^{(2)} \\ y_2^{(2)} \\ \vdots \\ y_m^{(2)} \\ \vdots \\ y_{(1)}^{(n)} \\ \vdots \\ y_{(m-1)}^{(n)} \\ y_{(m)}^{(n)} \end{bmatrix}_{nm\times 1} \tag{7}$$

Once trained, displacement ensemble was generated using 10 000 different input forces, and the DeepONet predictions were compared against the ground truth to get the RMSE. It can be observed that as the number of UIRs used to create the training dataset increases, the RMSE value decreases which is a direct effect of increased training data. In the same table, the performance of the trained model is also shown for cases where testing input forces are generated using different number of FT. As we increase the number of FT from 20 to 100 in prediction stage, the RMSE values increase, but they are still negligible for the case where training dataset is created using only 150 UIRs. This showcases ZSL capabilities of the surrogate model as it is able to produce good results for input distribution dissimilar to the one used for training.

Table 2
RMSE (m) values comparing performance of surrogate model for training datasets formed using different number of UIR, for Case-I (a). PPR are fixed at 100. Comparison is also drawn for case when tested for input corresponding to different number of FT. Note, that the training is done for 20 FT only.

R.M.S.E. (m)		FT				
		20	25	50	75	100
UIR	25	7.44E−04	8.53E−04	1.50E−03	2.02E−03	2.46E−03
	50	2.70E−04	3.01E−04	5.08E−04	7.68E−04	1.01E−03
	75	2.80E−04	3.44E−04	7.09E−04	1.00E−03	1.27E−03
	100	2.66E−04	2.99E−04	5.16E−04	7.42E−04	9.49E−04
	150	9.2E−05	1.04E−04	1.66E−04	2.65E−04	3.84E−04

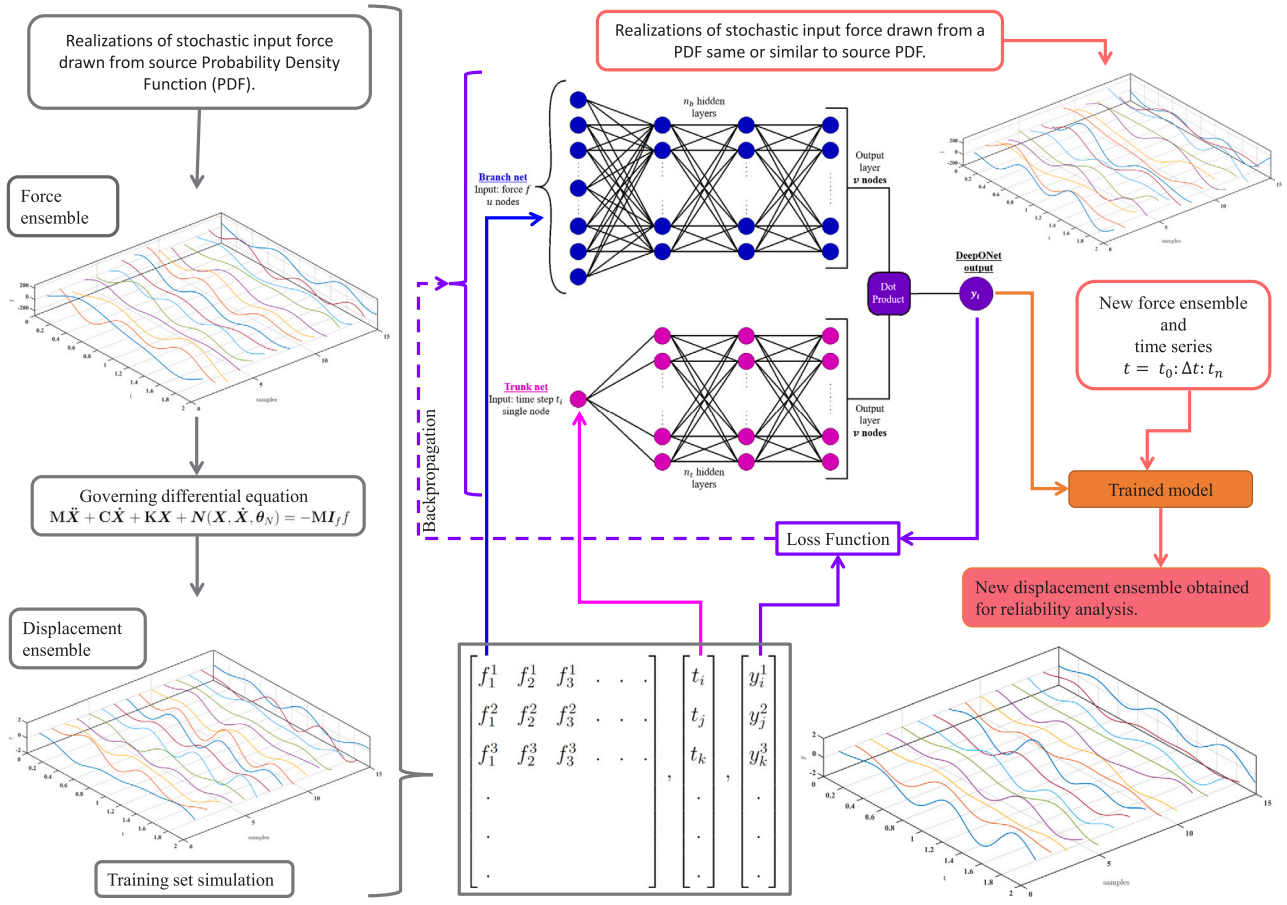


Fig. 4. Schematic showcasing the process flow adopted in this paper. From a source PDF, realizations of input forces are generated, and based on the governing differential equation; displacement ensemble is obtained. The training data thus generated is rearranged as shown in Eq. (3) and is passed through the DeepONet architecture. The trained model obtained from DeepONet is then introduced to new realizations of input force, and the corresponding displacements are generated with reduced computational cost. These displacements are then used to perform reliability analysis on the system under consideration.

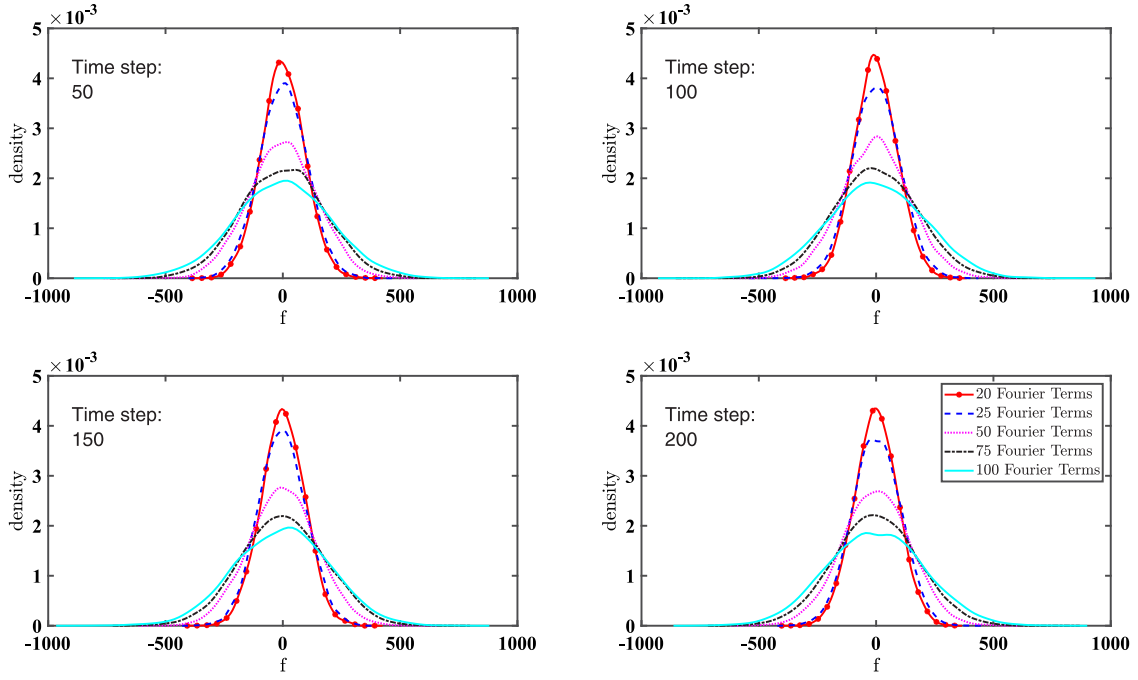


Fig. 5. Probability density functions for input force computed at any particular time step. Comparisons have been drawn for cases where input force ensemble is generated using different number of FT.

Table 3
RMSE (m) values comparing performance of surrogate model when trained for different number of PPR, for Case-I (a). UIRs in training dataset are fixed at 150. Comparison is also drawn for case when tested for input corresponding to different number of FT. Note, that the training is done for 20 FT only.

R.M.S.E. (m)		FT				
		20	25	50	75	100
PPR	1	2.47E−02	2.79E−02	4.12E−02	5.10E−02	5.89E−02
	25	3.02E−04	3.30E−04	5.14E−04	6.83E−04	9.34E−04
	50	1.56E−04	1.89E−04	4.07E−04	6.54E−04	9.04E−04
	75	3.91E−04	4.57E−04	8.28E−04	1.21E−03	1.58E−03
	100	1.29E−04	1.52E−04	3.30E−04	5.02E−04	7.01E−04

Figs. 6(a) and 6(b) shows the ensemble mean and variance respectively of 10 000 samples plotted against the ground truth when the DeepONet model is trained using dataset having 150 UIRs and 100 PPR. RNN and POD-GP were also trained for the dataset formed using 150 UIRs and complete displacement realizations were required to train both these models. The trained models were then tested using the same data as used for DeepONet and the results obtained were plotted for adequate comparison. It can be observed that while the mean values follow the ground truth closely in all the methods, DeepONet also captures the response variance better than the other two methods. The magnitude of variance in this case study is quite small as the behaviour of dynamical system under consideration is dominated by the initial conditions, which are a deterministic quantity. Learning such subtle variation for any data learning scheme is non-trivial, but still, DeepONet can follow the ground truth with reasonable accuracy. The plots further validate the trend shown by the RMSE values in Table 2 as the predicted values closely follow the ground truth.

Now that the variation in performance is tested for changing UIRs in the training dataset, another avenue of testing will be exploring the effects of changing PPR. Table 3 shows the RMSE values for varying PPR, keeping the number of UIRs in training dataset fixed at 150. It can be observed that as the number of PPR increases, the RMSE value decreases. This is simply due to the fact that while training, if PPR are more, surrogate model will get a better representation of the output function, even when UIRs in training dataset are kept same. Best performance among the tested PPR is observed for PPR equal to 100. It should be noted that in Tables 2 and 3, 50 UIRs with 100 PPR and 25 PPR with 150 UIRs in training dataset show negligible RMSE values. The choice of 150 UIRs in training dataset with 100 PPR however gave the best results for ZSL.

If the goal is only to perform predictions within the training distribution, fewer UIRs in training dataset and/or fewer PPR may also give good results.

4.1.2. Case-I (b)

To further showcase the flexibility and efficacy of DeepONet, for part *b*, input forces are taken as realizations of Gaussian process, which is defined as follows:

$$f \sim \mathcal{GP}(\mu(\boldsymbol{\theta}_\mu), \kappa(\boldsymbol{\theta}_\kappa)), \tag{8}$$

where $\mathcal{GP}(\cdot)$ is the Gaussian process with $\mu(\cdot)$ as its mean function and $\kappa(\cdot)$ as its kernel function. $\boldsymbol{\theta}_\mu$ and $\boldsymbol{\theta}_\kappa$ are the parameters for the mean and kernel function respectively. In this instance, f is taken as a zero mean Gaussian process with squared exponential function as its kernel,

$$\kappa(t,t') = \sigma^2 exp\left(-\frac{(t-t')^2}{2l^2}\right), \qquad \boldsymbol{\theta}_\kappa = [\sigma, l] \tag{9}$$

where σ is the variance parameter and l is the length scale parameter. Initial conditions for data simulation are taken as $x_0 = 0.005$ m, $\dot{x}_0 = 0.001$ m/s and $z_0 = 0.001$. For training the surrogate model variance parameter is taken as $\sigma = 50$ and the length scale parameter is taken as $l = 0.10$. 1000 UIRs with 50 PPR are taken for training the surrogate model.

Table 4
System parameters for Case-II: 5-DOF Duffing Oscillator.

Mass (kg)	Stiffness (N/m)	Damping (Ns/m)	Nonlinear parameters
$m_1 = 10, m_2 = 10$	$k_1 = 10\,000, k_2 = 10\,000$	$c_1 = 100, c_2 = 100$	$\alpha = 100$
$m_3 = 9, m_4 = 9$	$k_3 = 9000, k_4 = 9000$	$c_3 = 90, c_4 = 90$	
$m_5 = 7.5$	$k_5 = 7500$	$c_5 = 75$	

The surrogate model is tested for various combinations of σ and l and Fig. 7 shows the RMSE of 10 000 samples for each combination. It can be observed that while the RMSE values are higher for length scale below 0.10, they are of the order 10^{-4}. Similarly, for values of σ more than 50, RMSE values increase but stay within reasonable range. This shows the ZSL capabilities of the proposed framework. Equally good results were observed for 750 UIRs in training dataset but are not included in the interest of avoiding recapitulation of similar results.

Fig. 8 shows the predicted mean and variance for 10 000 test samples compared against the ground truth. As a detailed visual confirmation for the RMSE values plotted earlier, comparisons have been drawn for four different combinations of σ and l, and as expected, predictions closely follow the ground truth.

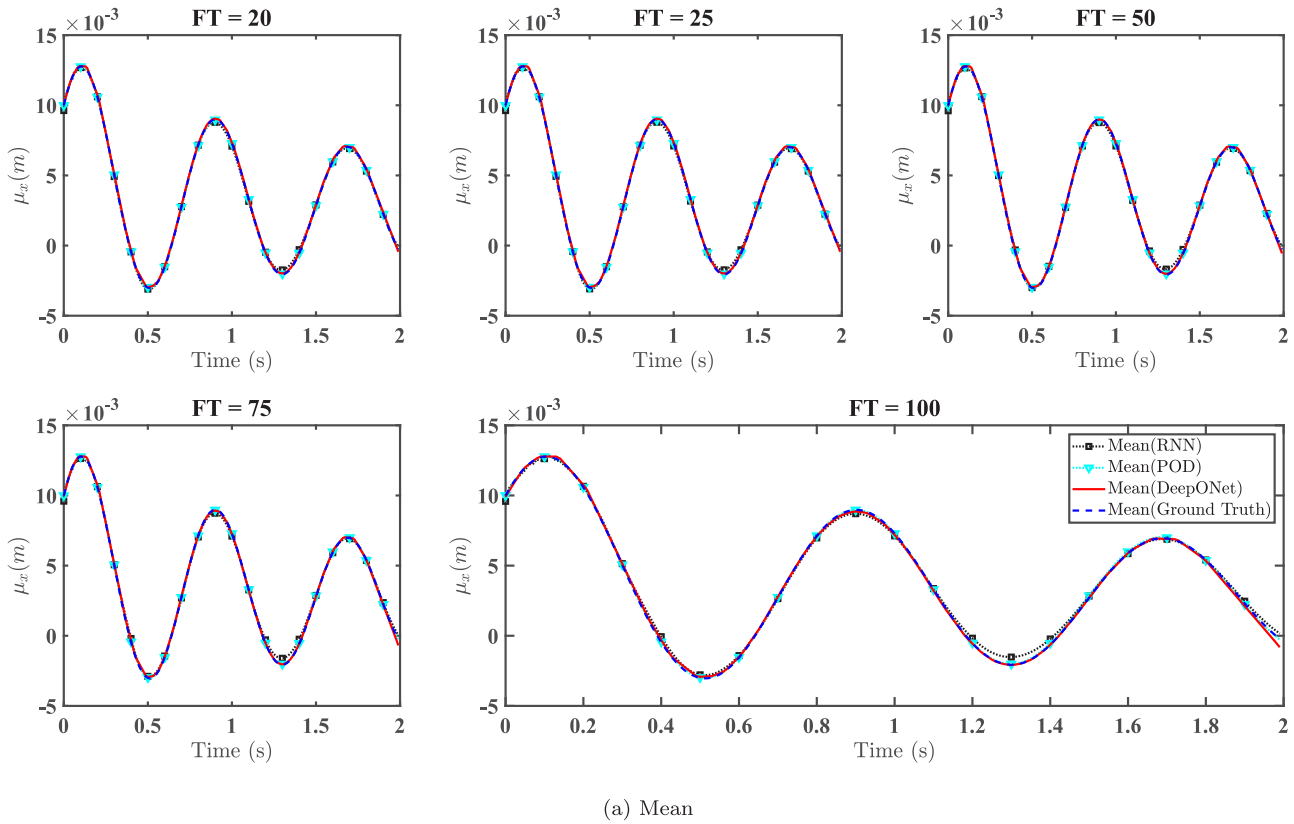
In this case study, the behaviour of the dynamical system was dominated by the initial conditions selected and the parameters of the nonlinear oscillator, but in the subsequent case studies, we will see the performance of the surrogate model for nonlinear multi DOF systems with non-zero initial conditions; whose behaviour is rather significantly affected by the input force. Also, since the maximum displacement is observed at initial time steps only, time dependent reliability analysis is skipped for this system.

4.2. Case-II: 5-DOF nonlinear system

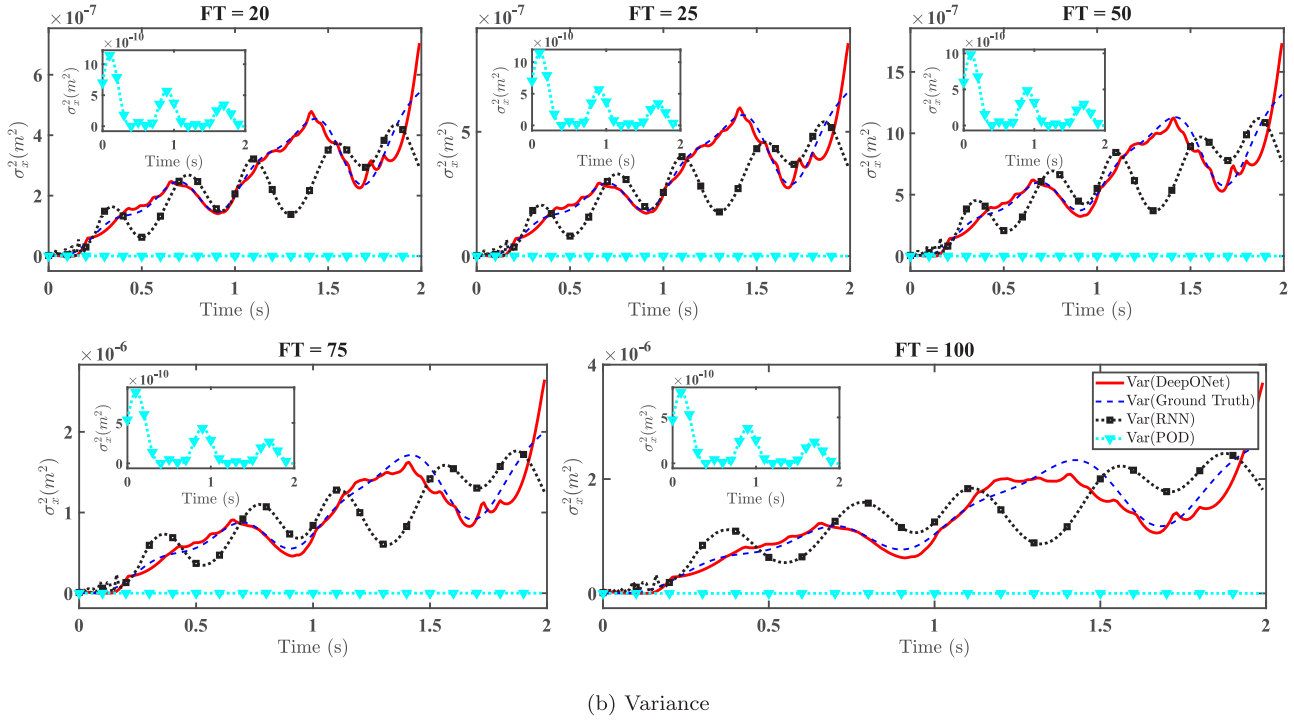
This case study deals with a base excited 5-DOF system with duffing oscillator installed at first DOF, governing equations of which are as follows:

$$\begin{gathered}
m_1\ddot{x}_1 + c_1\dot{x}_1 + c_2(\dot{x}_1 - \dot{x}_2) + k_1x_1 + k_2(x_1 - x_2) + \alpha_{do}x_1^3 = -m_1 f \\
m_2\ddot{x}_2 + c_2(\dot{x}_2 - \dot{x}_1) + c_3(\dot{x}_2 - \dot{x}_3) + k_2(\dot{x}_2 - x_1) + k_3(\dot{x}_2 - x_3) = -m_2 f \\
m_3\ddot{x}_3 + c_3(\dot{x}_3 - \dot{x}_2) + c_4(\dot{x}_3 - \dot{x}_4) + k_3(\dot{x}_3 - x_2) + k_4(\dot{x}_3 - x_4) = -m_3 f \\
m_4\ddot{x}_4 + c_4(\dot{x}_4 - \dot{x}_3) + c_5(\dot{x}_4 - \dot{x}_5) + k_4(\dot{x}_4 - x_3) + k_5(\dot{x}_4 - x_5) = -m_4 f \\
m_5\ddot{x}_5 + c_5(\dot{x}_5 - \dot{x}_4) + k_5(\dot{x}_5 - x_4) = -m_5 f, \\
x_i(0) = 0.01 \text{ m}, \ \dot{x}_i(0) = 0.05 \text{ m/s}, \ i = [1,2,3,4,5]
\end{gathered} \tag{10}$$

where m_i, c_i and k_i are the mass, damping and stiffness parameters respectively and α_{do} is the constant for duffing oscillator. f in this case study represents the ground acceleration, and randomness of input force is derived from the randomness of ground acceleration. Note that in Case-II and the engineering example, ground acceleration and input force have been used interchangeably because in both case studies, base excited systems have been considered. System parameters for the 5DOF system shown in Eq. (10) are given in Table 4. DeepONet architecture similar to previous case has been adopted but for multi DOF systems, during training, f is mapped to displacements in parallel i.e. for this case study 5 trained models will be available at the end of analysis. PPR are taken equal to 25 and performance of surrogate model is tested for 1500 UIRs in training dataset. Fig. 9 shows the performance of DeepONet, RNN and POD-GP models compared against the ground truth. Since the nonlinearity is reduced compared to the previous case study, the architecture for RNN was modified to extract 25 features from the RNN layer . Results produced for both RNN and DeepONet closely follows the ground truth. POD-GP however still fails to map the variance in the testing samples.



(a) Mean



(b) Variance

Fig. 6. Predicted mean (m) and variance (m^2) of 10 000 samples compared against the ground truth, for Case-I (a). Training dataset is formed using 150 UIRs and 100 PPR. Training has been done for input forces corresponding to 20 FT.

To show the ZSL capabilities of the DeepONet model, and compare the same against those of RNN and POD-GP, two realizations of input force corresponding to 25 FT and 50 FT were passed through the respective model. The subsequent results produced are plotted in Fig. 10 and it can be seen that performance of DeepONet is atleast on-par if not better than RNN. As for the comparison against POD-GP, DeepONet provides significant improvement in generalization for data which was not adequately represented during training stage.

Now that the predictions are available for a large number of testing samples, FPFT analysis can be performed. For FPFT analysis, thresholds

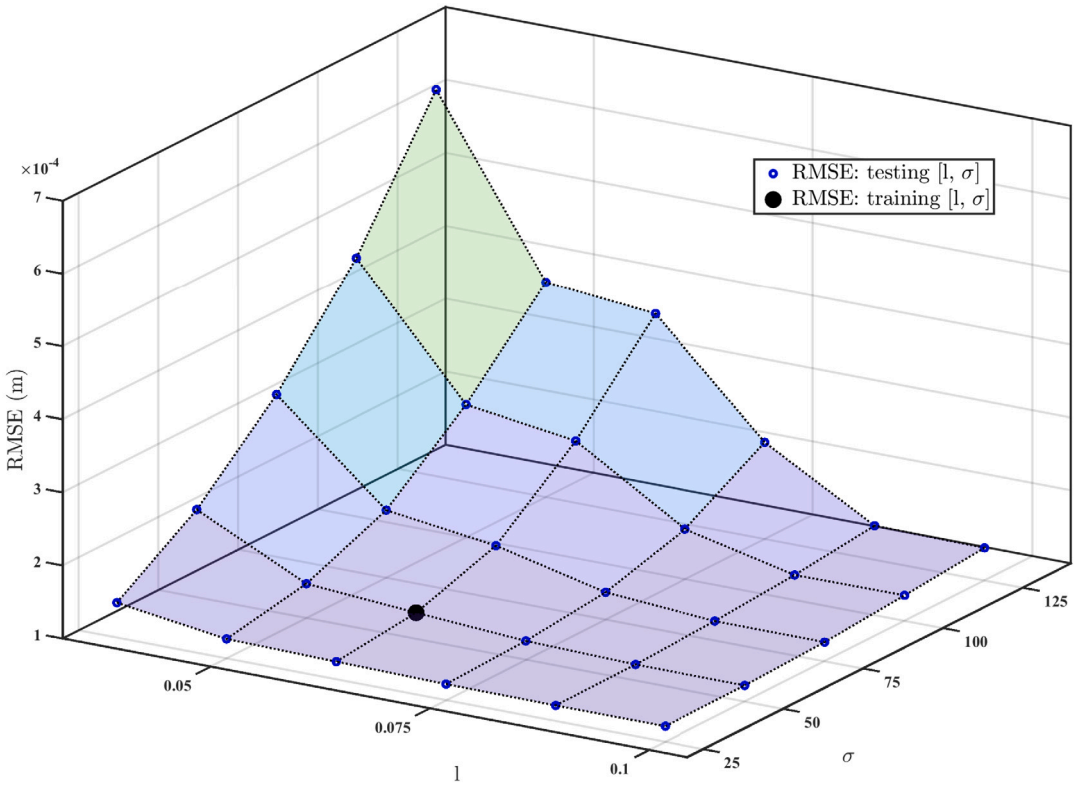


Fig. 7. RMSE (m) values comparing performance of surrogate model when tested for different combinations of σ and l, for Case-I (b). UIRs in training dataset are fixed at 1000 with 50 PPR. Note, that the training is done for $\sigma = 50$ and $l = 0.10$.

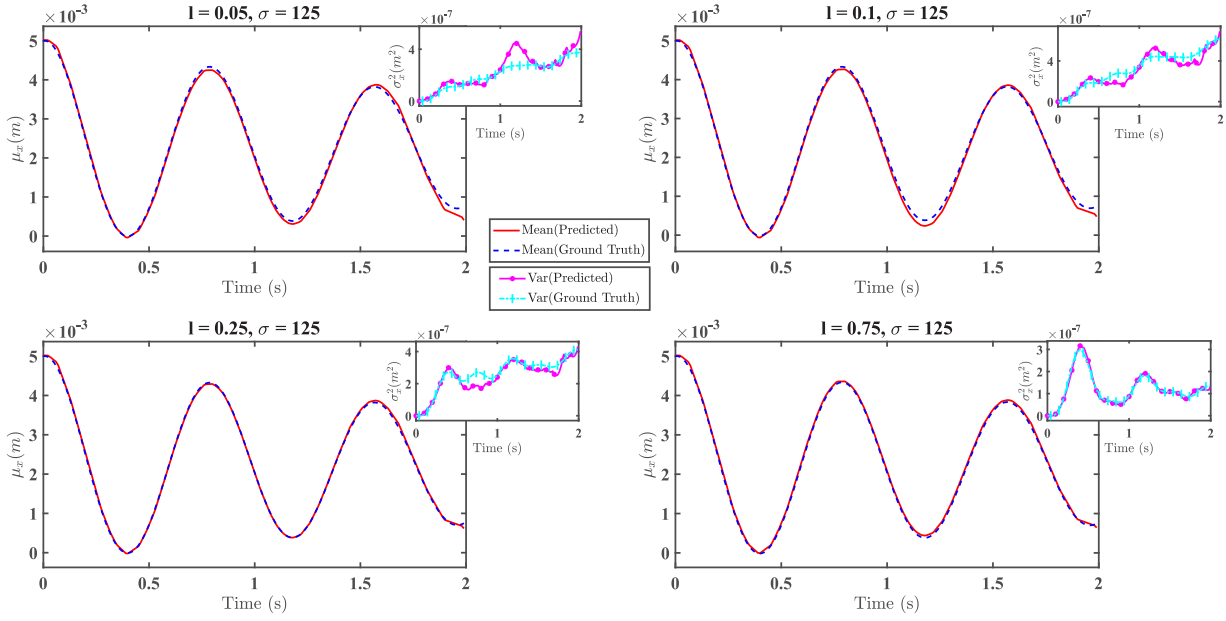


Fig. 8. Predicted mean (m) and variance (m^2) of 10 000 samples compared against the ground truth, for Case-I (b). 1000 UIRs are taken with 50 PPR to form the training dataset. Training is done for input forces corresponding to $\sigma = 50$ and $l = 0.10$. A snippet has been added showing the evolution of variance over time in all the subplots.

for maximum displacement were selected for each DOF and the time at which the respective displacement crossed the threshold was noted. For example, for 5th DOF, time at which $|x_5|$ crossed the magnitude of 2.5 was noted and the PDFs for FPFT were plotted. Fig. 11 shows the PDFs of FPFT for each DOF when the input force corresponds to 20 FT. It can be observed that the predicted PDFs are similar to the actual PDFs, thus demonstrating the applicability of the DeepONet surrogate model for reliability analysis.

Failure probabilities at different time steps have also been computed, limit state function $g(\cdot)$ for which is as follows:

$$g(t, \theta) = \mathcal{T}_i - max\{x_i(\tau, \theta); \tau \in [0, t]\}, \tag{11}$$

where $\mathcal{T}_i$ is the threshold and x_i is the displacement for ith DOF. θ in Eq. (11) represents the system parameters of the dynamical system. Probability of failure was computed using different surrogate models and the results are plotted in Fig. 12. It can be clearly seen that while

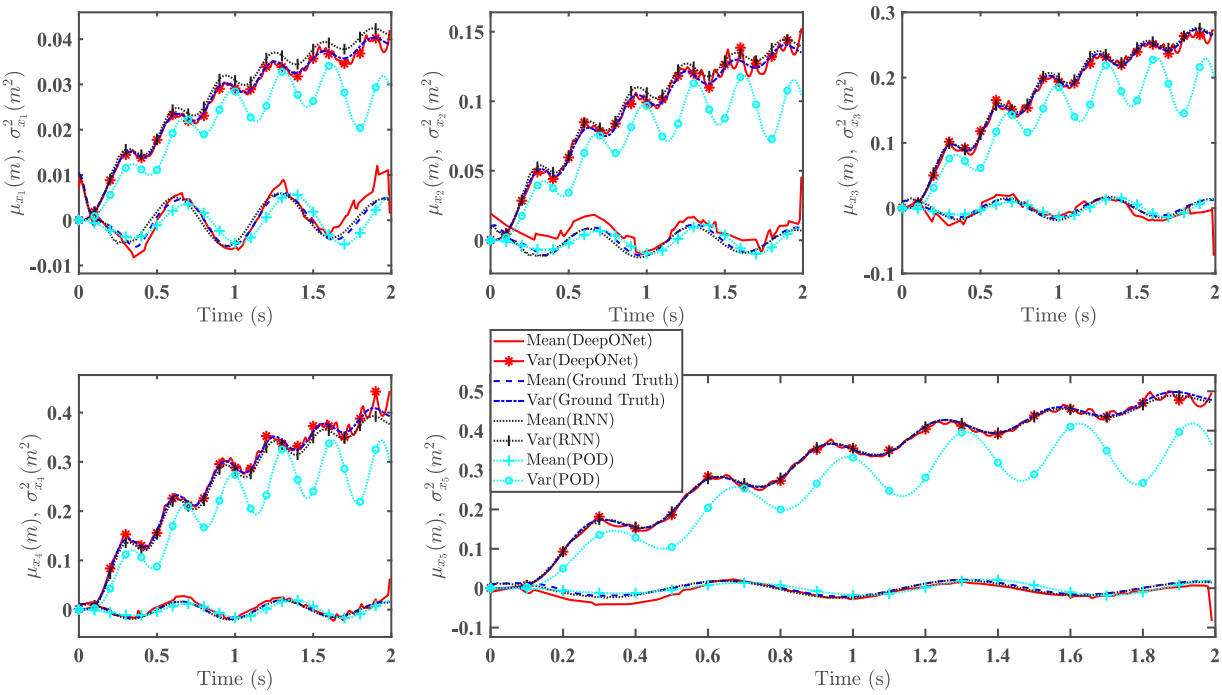


Fig. 9. Predicted mean (m) and variance (m^2) of 10 000 samples compared against the ground truth, for Case-II. 1500 UIRs are taken with 25 PPR to form the training dataset. Training and testing has been done for input forces corresponding to 20 FT.

RNN is an improvement over POD-GP, DeepONet comes closest to the ground truth. For reliability analysis, capturing the tail of the response PDF is essential for accurate analysis and here the performance of DeepONet is clearly better than that of the RNN.

This case study is a perfect scenario where if the user wishes to obtain displacement at Nth-DOF, they can use the trained DeepONet model to obtain just that without producing the results for all the other DOFs, thus saving the computational time as well as expensive server space. This is possible because displacement at each DOF is trained and mapped to input force individually. We note that this advantage holds for all surrogate models including the DeepONet model. However, such decoupling is not possible in classical time integration schemes. Another advantage of individually training the surrogate model is that if the RMSE at any particular DOF is high, UIRs or PPR in the training dataset for that particular DOF only can be changed, thus optimizing the computational requirements at the training stage.

5. Engineering example

The proposed DeepONet surrogate model is further tested for a benchmark problem presented in [60]. This case study showcases the scalability of the proposed approach by applying the concept of surrogate model to a base excited 76-DOF nonlinear system. The linear parameters for the dynamical system were selected as per the 76 storey ASCE benchmark [60,61], which was proposed as a office building in the city of Melbourne Australia. The 76-DOF building for the current case study is modified to include a Bouc Wen oscillator at first DOF. The 306.1 m high building has a concrete frame of size 42 m × 42 m and a concrete core of size 21 m × 21 m. Square columns are given at the periphery of the concrete frame, 6.5 m apart and are connected by a spandrel beam at each floor. A schematic for the building plan and elevation is shown in Fig. 13.

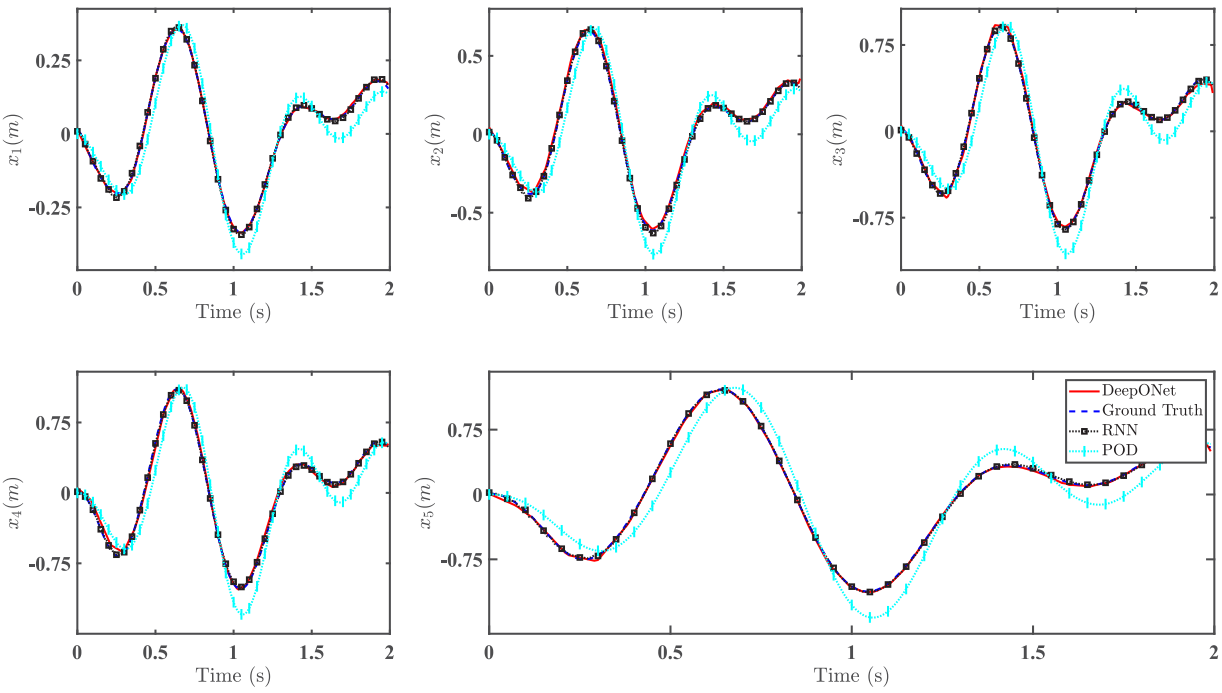
Appropriate changes have been made in the differential equations of the system to accommodate the Bouc-Wen oscillator. For data simulation using Runge-Kutta scheme, a sampling frequency of 12 500 Hz was adopted, but data provided for training was sampled at a frequency of 100 Hz. The rest of the DeepONet architecture was kept similar to previous case studies, with UIRs in training dataset here being 400 with 100 PPR. Similar to the previous MDOF case study, f will be mapped in parallel to displacements, and hence 76 different trained models will be available. This brings with it the advantage of training displacements at only required DOFs, thus saving computational cost and time when compared against conventional MCMC based techniques. Fig. 14 shows the percentage Normalized Mean Square Error(NMSE) for 10 000 samples at each DOF. NMSE has been calculated as follows:

$$NMSE = \frac{1}{N_s}\sum_{i=1}^{N_s}\frac{\|predicted - actual\|_2^2}{\|actual\|_2^2}, \tag{12}$$

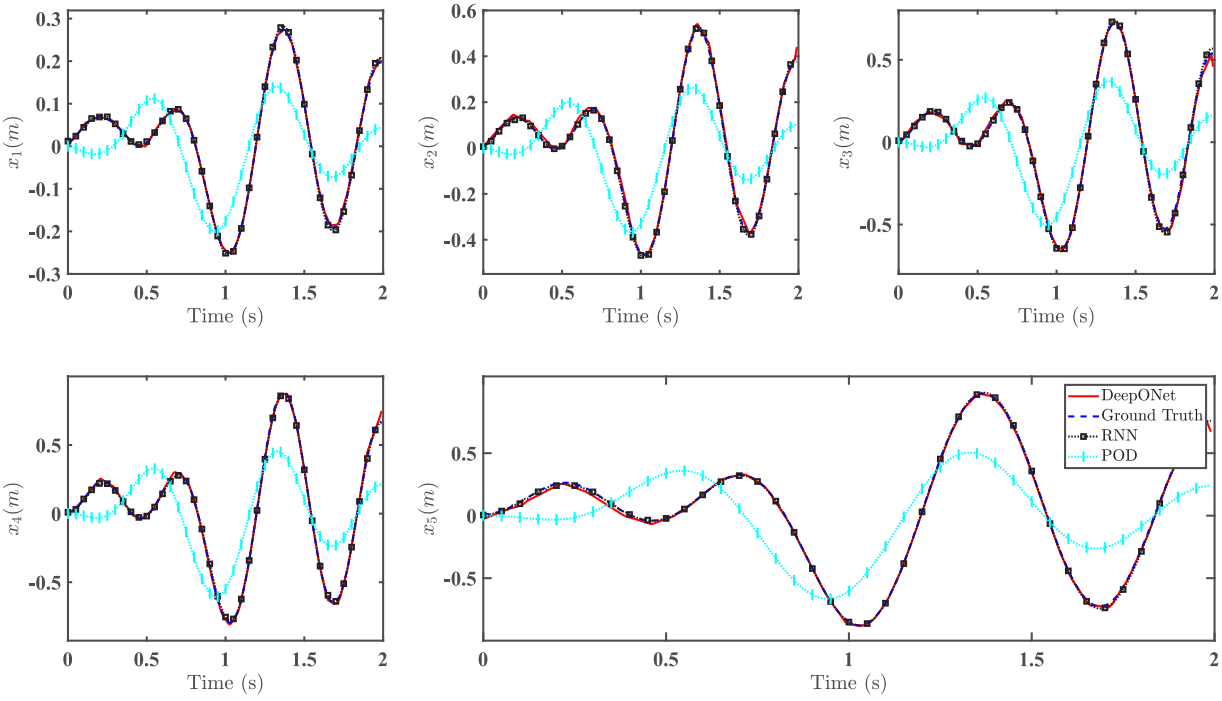
The results produced show that the NMSE is well below 1% thus giving a reasonably accurate picture of the actual system.

To simulate a practical scenario, predictions were made for a realization of force generated using Kanai-Tajimi earthquake model [62]. The surrogate model for predictions is trained for the input forces corresponding to Eq. (5); 20 FTs were considered for training input. The predictions corresponding to the Kanai-Tajimi earthquake force are plotted in Fig. 15. DeepONet results obtained are close to ground truth while RNN fails to converge as we move towards the top of the building; this is perhaps because RNN suffered with the issue of vanishing gradients, as indicated by the fact that the training loss was not reducing beyond a certain value.

Similar to previous case studies, FPFT PDFs have been computed using the DeepONet and RNN surrogate models and the same are plotted in Fig. 16. Results are shown for 5th, 15th, 35th, 65th and 75th DOF. The predicted PDFs obtained using DeepONet surrogate model closely follow the ground truth, thus enabling the designer to make well-informed decisions. Failure probabilities for limit state function defined in Eq. (11) at different time steps have been plotted in Fig. 17. It can be clearly seen that for the current dynamical system, performance of DeepONet is far better than that of RNN. Note, that the displacements produced for this case study, 76-DOF buildings have high magnitudes for top storeys as no control structure have been included in the analysis and despite this the surrogate model is still able to predict the displacements with great accuracy.



(a) Input force corresponding to 25FT.



(b) Input force corresponding to 50FT.

Fig. 10. Predicted displacement (m) for input realizations corresponding to 25 FT and 50 FT. Training is done for input forces corresponding to 20 FT.

6. Conclusion

Reliability analysis is an integral part of any structural design, especially for structures subjected to uncertain loading conditions, and to perform a robust reliability analysis, ample data representing the actual system is often needed. To achieve this, help of computer aided simulations is taken, as it is possible to test a wide variety of cases with little modification to underlying code. But generating high-fidelity data has associated with it immense computational cost, especially for complex nonlinear dynamical systems. In this paper, we propose a surrogate model based technique to perform reliability analysis using neural networks, specifically DeepONet architecture. Unlike conventional machine learning models, DeepONet learns a function to function mapping and hence, is ideally suited for time-dependent reliability analysis and uncertainty quantification problems subjected to loading uncertainty. The proposed framework enables efficient reliability analysis, and also provides ZSL capabilities. Using the proposed framework, it is possible

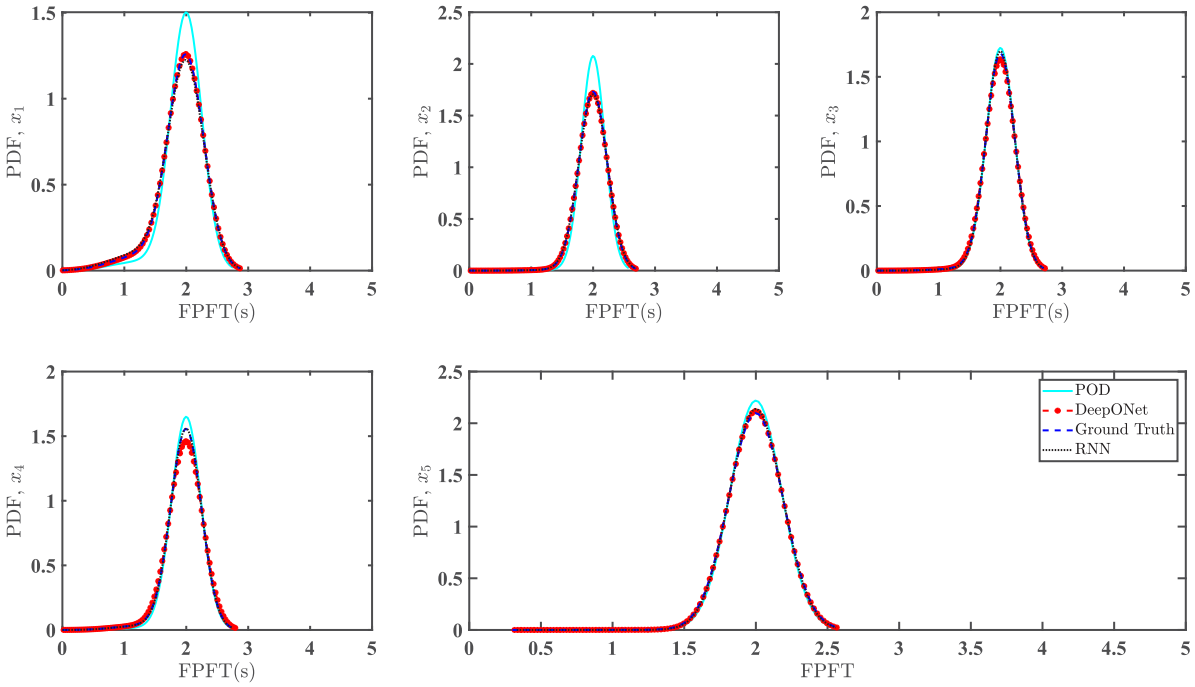


Fig. 11. PDFs of FPFT obtained for Case-II. Training and testing are done for input forces corresponding to 20 FT.

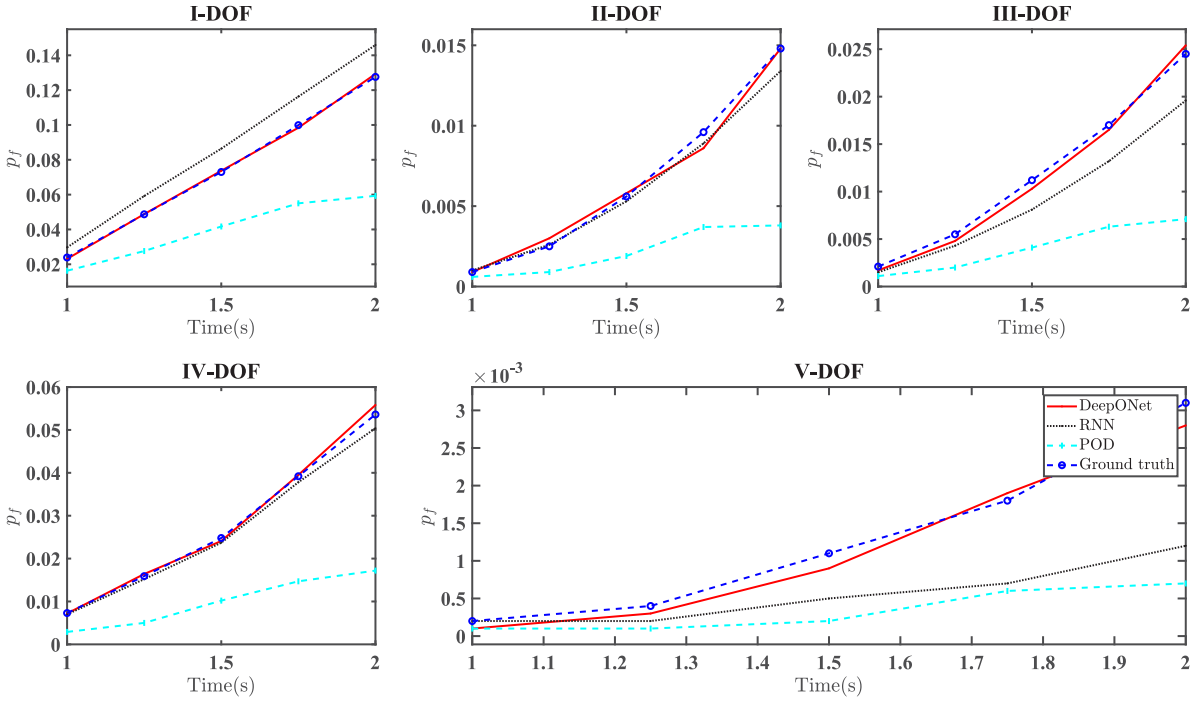


Fig. 12. Failure probability computed at different time steps for Case-II. Training and testing are done for input forces corresponding to 20 FT.

to predict only the required data at coarser time steps, thus saving both expensive server space and computational cost. Authors note that time will be spent on training the surrogate model, but the time saved in the prediction stage will sufficiently compensate for the time spent in training. The proposed framework will especially benefit applications where the data simulation takes enormous amounts of time.

As a proof of concept, several case studies have been discussed above, testing the surrogate model against a variety of dynamical systems and assessing the use of DeepONet architecture for reliability analysis of nonlinear dynamical systems. Case-I and II demonstrate the surrogate model's capability to model nonlinear oscillators, while the engineering example covered showcase the scalability of the proposed framework. Although at this stage, the predictions have been made for 10^4 samples, in a practical scenario, 10^5 or even more test samples may be required, and in those cases, the proposed algorithm can be immensely helpful in saving simulation time. The results produced in the case studies represent the actual system with good accuracy. Input realizations used to form the training dataset were kept limited so as to showcase the performance of surrogate model with less training data,

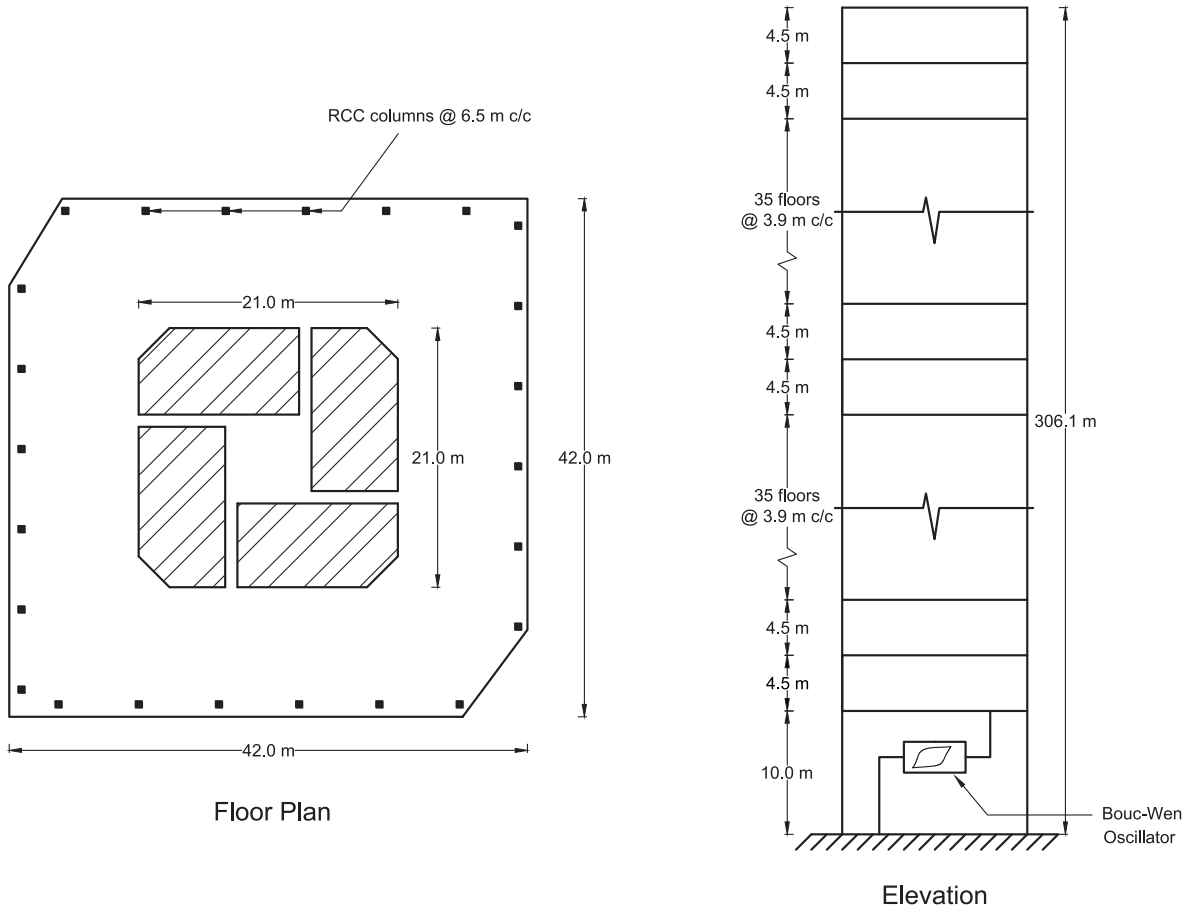


Fig. 13. Schematics for the 76-DOF ASCE benchmark building augmented with Bouc-Wen oscillator at first DOF. The benchmark presented in [60] is a 76-storey office building proposed for the city of Melbourne, Australia.

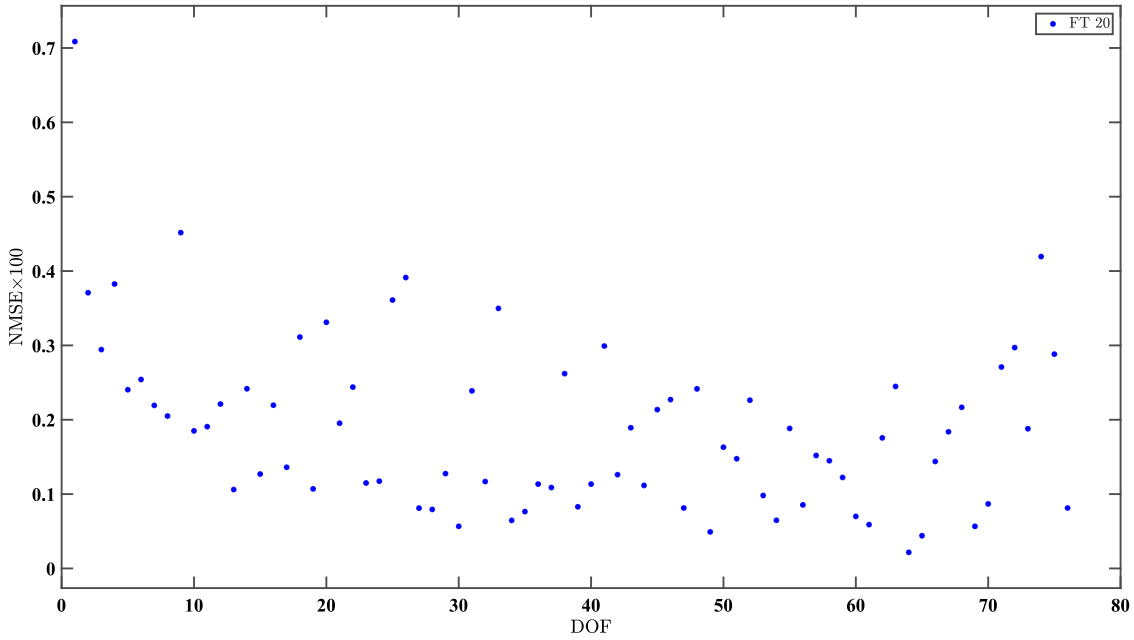


Fig. 14. Percentage NMSE for 10 000 samples (engineering case study). Model has been trained and tested for 20 FT.

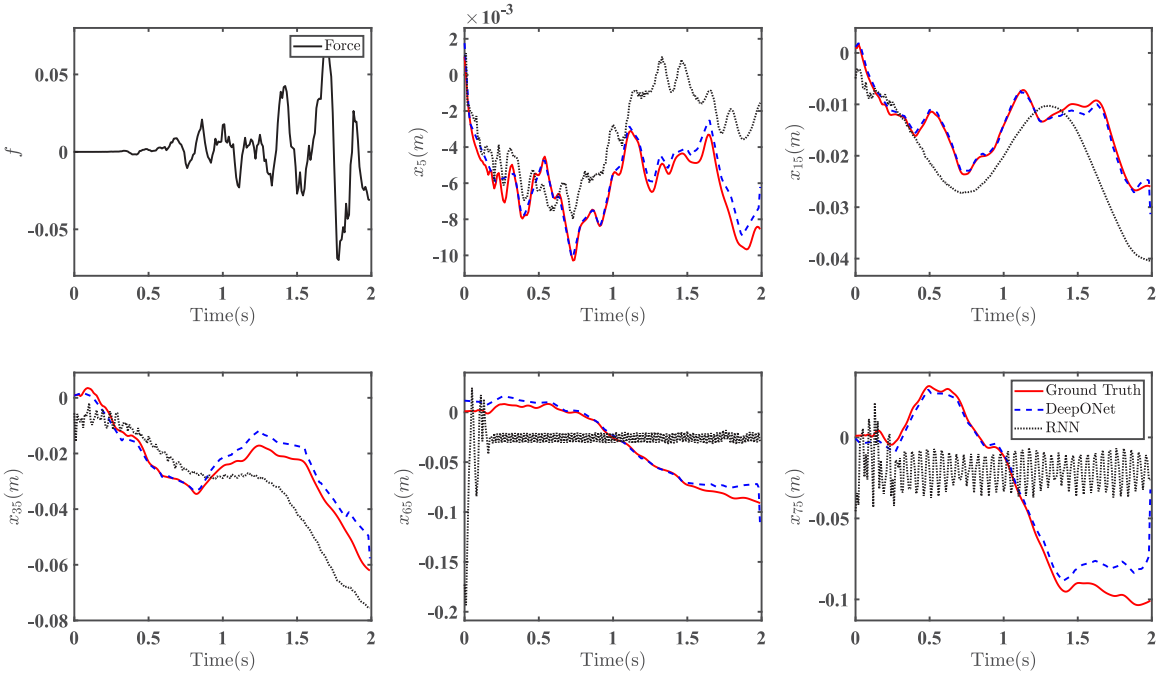


Fig. 15. Predicted displacement (m) compared against the ground truth, for 76-DOF nonlinear building. The input force realization here was simulated to reflect the earthquake ground motion. Results are shown at 5th, 15th, 35th, 65th and 75th DOF. Training has been done for input forces corresponding to 20 FT generated using Eq. (5).

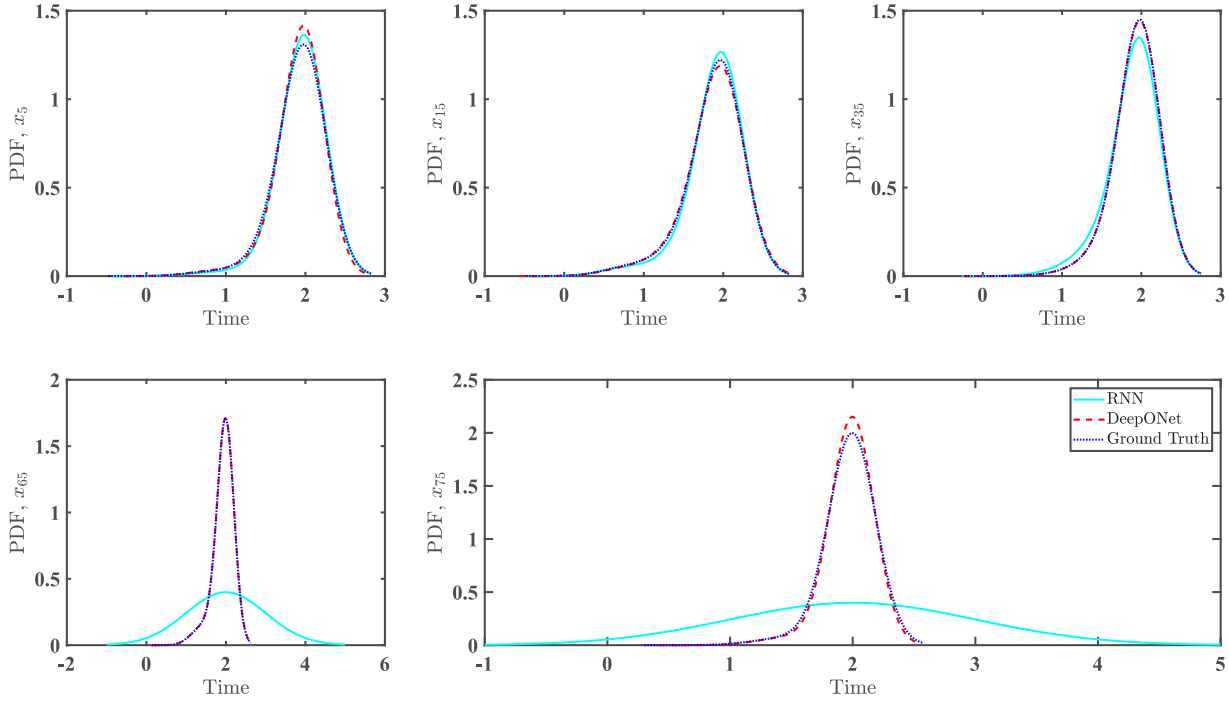


Fig. 16. PDFs of FPFT obtained for engineering case study at 5th, 15th, 35th, 65th and 75th DOF. Training and testing are done for input forces corresponding to 20FT.

and from the case studies, it can be observed that good results are obtained.

Based on the requirements of the system under consideration, DeepONet architecture can be easily tweaked, and appropriate results can be obtained. Some key points to note include the following. PPR selected should be uniformly distributed in the domain of the function for best training results. Number of nodes and hidden layers may be increased if the length of individual samples in the training data are large. As a rule of thumb, we recommend increasing the number of hidden layers first, before moving on to nodes per layer as this approach mitigates some chances of over-fitting. Apart from this, importance of normalization in any NN architecture cannot be overstated, and hence the data being given as the training input should be appropriately normalized. Although the choice of normalization will depend on the application and type of data, we propose using Keras *preprocessing normalization layer* [63]. Min–max scaling function was also tested, and satisfactory results were obtained.

Having shown the efficacy of the DeepONet framework, authors note the fact that at the current stage, DeepONet's use is limited to problems with uncertainty in input domain and further modifications

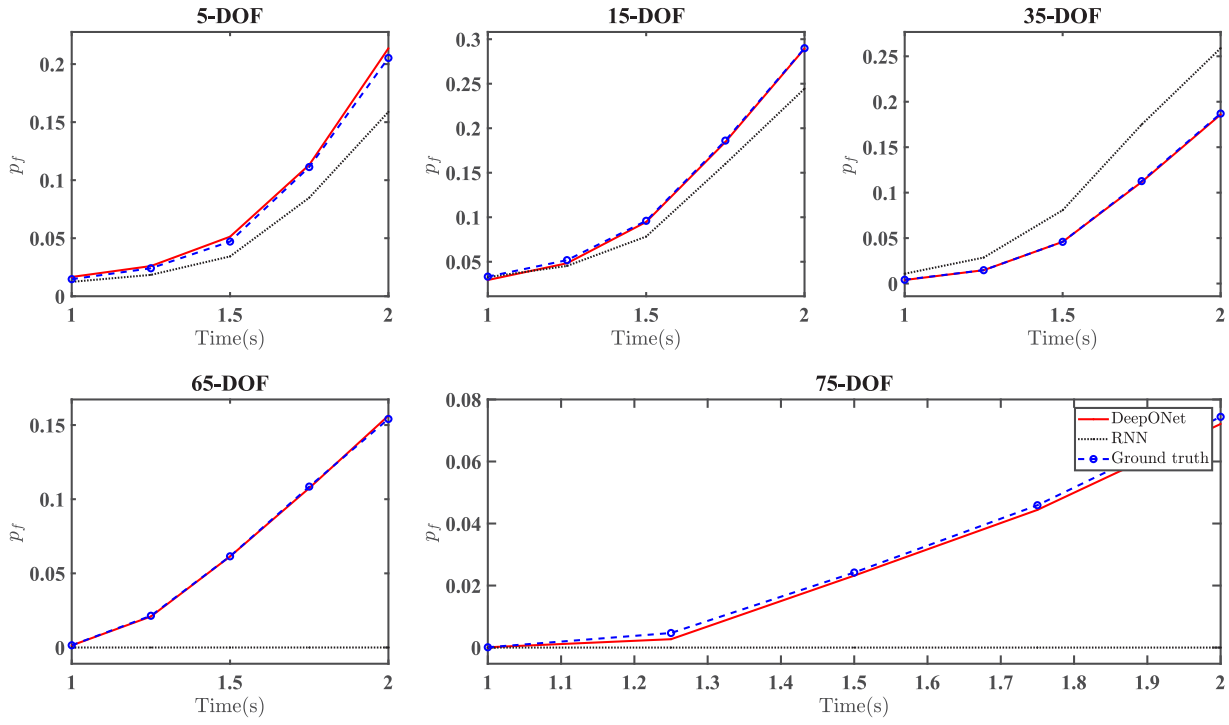


Fig. 17. Probability of failure at different time points, obtained for engineering case study at 5th, 15th, 35th, 65th and 75th DOF. Training and testing are done for input forces corresponding to 20FT.

will be required to the proposed framework to analyse dynamical systems with parametric uncertainty. Authors would also like to acknowledge that while the goal of this paper was to access the use of DeepONet architecture for analysis of dynamical systems subjected to stochastic forces, other Deep NN architectures may also hold the capacity to act as a surrogate model, a combined analysis of which is a behemoth task left open for research.

CRediT authorship contribution statement

Shailesh Garg: Formal analysis, Data curation, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Harshit Gupta:** Software, Validation, Visualization. **Souvik Chakraborty:** Conceptualization, Funding acquisition, Project administration, Resources, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The codes and the data associated with this paper can be accessed at https://github.com/csccm-iitd/DeepONet-reliability.

Acknowledgements

SG acknowledges the support received from the Ministry of Education in the form of Ph.D. scholarship. SC acknowledges the financial support received from Science and Engineering Research Board (SERB) vi grant no. SRG/2021/000467 and from IIT Delhi in the form of seed grant.

References

[1] Rackwitz R. Reliability analysis—a review and some perspectives. Struct Saf 2001;23(4):365–95.
[2] Melchers RE, Beck AT. Structural reliability analysis and prediction. John wiley & sons; 2018.
[3] Ditlevsen O, Madsen HO. Structural reliability methods, Vol. 178. Wiley New York; 1996.
[4] Rüemelin W. Numerical treatment of stochastic differential equations. SIAM J Numer Anal 1982;19(3):604–13.
[5] Morton KW, Mayers DF. Numerical solution of partial differential equations: an introduction. Cambridge University Press; 2005.
[6] Butcher JC. A history of Runge-Kutta methods. Appl Numer Math 1996;20(3):247–60.
[7] Subbaraj K, Dokainish M. A survey of direct time-integration methods in computational structural dynamics—II. Implicit methods. Comput Struct 1989;32(6):1387–401.
[8] Paz M. Structural dynamics: theory and computation. Springer Science & Business Media; 2012.
[9] Rubinstein RY, Kroese DP. Simulation and the Monte Carlo method, Vol. 10. John Wiley & Sons; 2016.
[10] Engelund S, Rackwitz R. A benchmark study on importance sampling techniques in structural reliability. Struct Saf 1993;12(4):255–76.
[11] Papaioannou I, Papadimitriou C, Straub D. Sequential importance sampling for structural reliability analysis. Struct Saf 2016;62:66–75.
[12] Helton JC, Davis FJ. Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. Reliab Eng Syst Saf 2003;81(1):23–69.
[13] Au S-K, Beck JL. Estimation of small failure probabilities in high dimensions by subset simulation. Probab Eng Mech 2001;16(4):263–77.
[14] Chakraborty S, Tesfamariam S. Subset simulation based approach for space-time-dependent system reliability analysis of corroding pipelines. Struct Saf 2021;90:102073.
[15] Wold S, Esbensen K, Geladi P. Principal component analysis. Chemometr Intell Lab Syst 1987;2(1–3):37–52.
[16] Comon P. Independent component analysis. 1992.
[17] Hesthaven JS, Ubbiali S. Non-intrusive reduced order modeling of nonlinear problems using neural networks. J Comput Phys 2018;363:55–78.
[18] Schilders WH, Van der Vorst HA, Rommes J. Model order reduction: theory, research aspects and applications, Vol. 13. Springer; 2008.
[19] Mohri M, Rostamizadeh A, Talwalkar A. Foundations of machine learning. MIT Press; 2018.

[20] Jordan MI, Mitchell TM. Machine learning: Trends, perspectives, and prospects. Science 2015;349(6245):255–60.
[21] Liu Y, Zhao T, Ju W, Shi S. Materials discovery and design using machine learning. J Mater 2017;3(3):159–77.
[22] Wang W, Siau K. Artificial intelligence, machine learning, automation, robotics, future of work and future of humanity: A review and research agenda. J Database Manage (JDM) 2019;30(1):61–79.
[23] Worden K, Manson G. The application of machine learning to structural health monitoring. Phil Trans R Soc A 2007;365(1851):515–37.
[24] Wuest T, Weimer D, Irgens C, Thoben K-D. Machine learning in manufacturing: advantages, challenges, and applications. Prod Manuf Res 2016;4(1):23–45.
[25] Sarle W. Neural networks and statistical models. In: Proceedings of the 19th annual SAS users group international conference. SAS Institute; 1994, p. 1538–50.
[26] Gurney K. An introduction to neural networks. CRC Press; 2018.
[27] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 2019;378:686–707.
[28] Silvescu A. Fourier neural networks. In: IJCNN'99. international joint conference on neural networks. proceedings (Cat. No. 99CH36339), Vol. 1. IEEE; 1999, p. 488–91.
[29] Lu L, Jin P, Pang G, Zhang Z, Karniadakis GE. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nat Mach Intell 2021;3(3):218–29.
[30] Chen T, Chen H. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. IEEE Trans Neural Netw 1995;6(4):911–7.
[31] Arnold L. Stochastic differential equations. New York; 1974.
[32] Xian Y, Lampert CH, Schiele B, Akata Z. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. IEEE Trans Pattern Anal Mach Intell 2018;41(9):2251–65.
[33] Atkinson S, Zabaras N. Structured Bayesian Gaussian process latent variable model: Applications to data-driven dimensionality reduction and high-dimensional inversion. J Comput Phys 2019;383:166–95.
[34] Chakraborty S, Chowdhury R. Graph-theoretic-approach-assisted Gaussian process for nonlinear stochastic dynamic analysis under generalized loading. J Eng Mech 2019;145(12):04019105.
[35] Manfredi P, Trinchero R. A probabilistic machine learning approach for the uncertainty quantification of electronic circuits based on Gaussian process regression. IEEE Trans Comput-Aided Des Integr Circuits Syst 2021.
[36] Chakraborty S. Transfer learning based multi-fidelity physics informed deep neural network. J Comput Phys 2021;426:109942.
[37] Chakraborty S, Chowdhury R. Moment independent sensitivity analysis: H-PCFE–based approach. J Comput Civ Eng 2017;31(1):06016001.
[38] Chakraborty S, Chowdhury R. Polynomial correlated function expansion. In: Modeling and simulation techniques in structural engineering. IGI global; 2017, p. 348–73.
[39] Chakraborty S, Chatterjee T, Chowdhury R, Adhikari S. A surrogate based multi-fidelity approach for robust design optimization. Appl Math Model 2017;47:726–44.
[40] Sudret B. Meta-models for structural reliability and uncertainty quantification. 2012, arXiv preprint arXiv:1203.2062.
[41] Sudret B. Surrogate models for efficient uncertainty quantification. In: Euromech colloquium 618: uncertainty quantification in computational mechanics. ETH Zurich, Chair of Risk, Safety and Uncertainty Quantification; 2021.
[42] Chowdhury R, Adhikari S. Fuzzy parametric uncertainty analysis of linear dynamical systems: A surrogate modeling approach. Mech Syst Signal Process 2012;32:5–17.
[43] Pulch R, Youssef M. Machine learning for trajectories of parametric nonlinear dynamical systems. J Mach Learn Model Comput 2020;1(1).
[44] Yaghoubi V, Marelli S, Sudret B, Abrahamsson T. Sparse polynomial chaos expansions of frequency response functions using stochastic frequency transformation. Probab Eng Mech 2017;48:39–58.
[45] Manfredi P, Trinchero R. A data compression strategy for the efficient uncertainty quantification of time-domain circuit responses. IEEE Access 2020;8:92019–27.
[46] Connor JT, Martin RD, Atlas LE. Recurrent neural networks and robust time series prediction. IEEE Trans Neural Netw 1994;5(2):240–54.
[47] Medsker LR, Jain L. Recurrent neural networks. Des Appl 2001;5:64–7.
[48] Chatterjee A. An introduction to the proper orthogonal decomposition. Current Sci 2000;808–17.
[49] Jacquelin E, Baldanzini N, Bhattacharyya B, Brizard D, Pierini M. Random dynamical system in time domain: A POD-PC model. Mech Syst Signal Process 2019;133:106251.
[50] Lutes LD, Sarkani S. Reliability analysis of systems subject to first-passage failure. 2009.
[51] Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, et al. Tensorflow: A system for large-scale machine learning. In: 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16). 2016, p. 265–83.
[52] Wen Y-K. Method for random vibration of hysteretic systems. J Eng Mech Div 1976;102(2):249–63.
[53] Li H-g, Meng G. Nonlinear dynamics of a SDOF oscillator with Bouc–Wen hysteresis. Chaos Solitons Fractals 2007;34(2):337–43.
[54] Agarap AF. Deep learning using rectified linear units (relu). 2018, arXiv preprint arXiv:1803.08375.
[55] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings; 2010, p. 249–56.
[56] Cazemier W, Verstappen R, Veldman A. Proper orthogonal decomposition and low-dimensional models for driven cavity flows. Phys Fluids 1998;10(7):1685–99.
[57] Chen L, Letchford CW. Proper orthogonal decomposition of two vertical profiles of full-scale nonstationary downburst wind speeds [lzcl]. J Wind Eng Ind Aerodyn 2005;93(3):187–216.
[58] Iuliano E, Quagliarella D. Proper orthogonal decomposition, surrogate modelling and evolutionary optimization in aerodynamic design. Comput & Fluids 2013;84:327–50.
[59] Shirali S. Principal Component and independent component regression for predicting the responses of nonlinear base isolated structures. (Masters thesis), University of Waterloo; 2009.
[60] Yang JN, Agrawal AK, Samali B, Wu J-C. Benchmark problem for response control of wind-excited tall buildings. J Eng Mech 2004;130(4):437–46.
[61] Nayek R, Chakraborty S, Narasimhan S. A Gaussian process latent force model for joint input-state estimation in linear structural systems. Mech Syst Signal Process 2019;128:497–530.
[62] Cheynet E. Earthquake simulation and fitting. 2020, http://dx.doi.org/10.5281/ZENODO.3774070, URL: https://zenodo.org/record/3774070.
[63] tf.keras.layers. normalization. 2021, URL: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Normalization Accessed: 10 Jan 2022, 0040 Hrs.