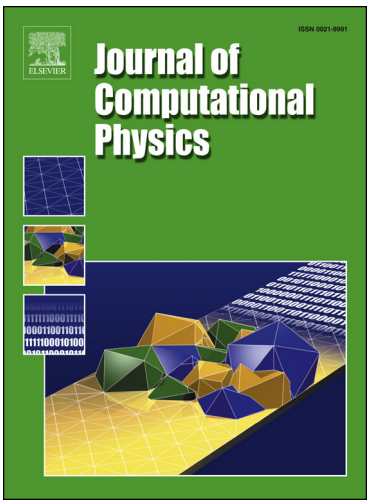
# Journal Pre-proof




Transfer learning based multi-fidelity physics informed deep neural network

Souvik Chakraborty

PII: S0021-9991(20)30716-6
DOI: https://doi.org/10.1016/j.jcp.2020.109942
Reference: YJCPH 109942

To appear in: *Journal of Computational Physics*

Received date: 19 May 2020
Revised date: 9 September 2020
Accepted date: 19 October 2020

Please cite this article as: S. Chakraborty, Transfer learning based multi-fidelity physics informed deep neural network, *Journal of Computational Physics*, 109942, doi: https://doi.org/10.1016/j.jcp.2020.109942.

## Highlights

- Multi-fidelity physics informed deep neural network (MF-PIDNN) is proposed.
- MF-PIDNN blends physics-informed and data-driven deep neural networks.
- MF-PIDNN is suitable when governing equation is approximately known.
- MF-PIDNN needs no low-fidelity data
- Results presented illustrate the accuracy of MF-PIDNN.

# Transfer learning based multi-fidelity physics informed deep neural network

Souvik Chakraborty[a,*]

[a]*Department of Applied Mechanics, Indian Institute of Technology Delhi, Hauz Khas - 110016, India.*

**Abstract**

For many systems in science and engineering, the governing differential equation is either not known or known in an approximate sense. Analyses and design of such systems are governed by data collected from the field and/or laboratory experiments. This challenging scenario is further worsened when data-collection is expensive and time-consuming. To address this issue, this paper presents a novel multi-fidelity physics informed deep neural network (MF-PIDNN). The framework proposed is particularly suitable when the physics of the problem is known in an approximate sense (low-fidelity physics) and only a few high-fidelity data are available. MF-PIDNN blends physics informed and data-driven deep learning techniques by using the concept of transfer learning. The approximate governing equation is first used to train a low-fidelity physics informed deep neural network. This is followed by transfer learning where the low-fidelity model is updated by using the available high-fidelity data. MF-PIDNN is able to encode useful information on the physics of the problem from the *approximate* governing differential equation and hence, provides accurate prediction even in zones with no data. Additionally, no low-fidelity data is required for training this model. Two examples involving function approximations with linear and nonlinear correlation are presented to illustrate the effectiveness of transfer learning in solving multi-fidelity problems. Applicability and utility of MF-PIDNN are illustrated in solving four benchmark reliability analysis problems. Case studies presented illustrate interesting features of the proposed approach.

*Keywords:* multi-fidelity, deep learning, physics-informed, transfer learning, reliability

## 1. Introduction

The governing equations used in science and engineering are often based on certain assumptions and approximations [1]. For example, heterogeneous material properties are approximated as ho-

*Corresponding authors

*Email address:* souvik@am.iitd.ac.in (Souvik Chakraborty)

mogeneous [2], effect of environmental conditions are rarely considered [3] and critical parts such as joints are often ignored [4]. Naturally, results obtained by solving the governing equations only provide an approximation of the actual system behavior (i.e., low-fidelity results). An alternative is to perform actual experiments in a laboratory environment. With modern experimental setups and sensors, it is possible to perform highly sophisticated experiments [5, 6]. Results obtained from such experiments are generally accurate (high-fidelity results). However, experiments are expensive and time-consuming, and one can only perform a limited number of experiments (usually in the order of tens). Such a small number of experiments is often not sufficient for understanding the system behavior, specifically if dealing with problems such as uncertainty quantification and reliability analysis. Furthermore, owning to the fact that the two data-sets are of different fidelity, merging the two data set into one is not an option.

One possible solution to the difficulties raised above resides in multi-fidelity schemes [7–9] where both high-fidelity and low-fidelity data can be used. The most popular multi-fidelity schemes are perhaps the multi-level Monte Carlo (MLMC) methods [10–13]. The primary idea in MLMC is to accelerate the calculation of the second moments of the quantity of interests. Another popular approach for dealing with multi-fidelity data is co-Kriging [14–17]. In this method, Kriging [18–21], aka Gaussian process [22–26], is coupled with an auto-regressive like information fusion scheme [27–29]. Methods where the Gaussian process in co-Kriging is replaced by other machine learning techniques can also be found in the literature [30–33]. The success of all these methods is already well-established in the literature [34, 35]. Unfortunately, these methods only work for cases where the low-fidelity data is able to capture the trend and the models of different fidelities have a strong linear correlation Both co-Kriging motivated approaches and MLMC fails when the low-fidelity and high-fidelity data have a space-dependent, complex and nonlinear correlations. To address this issue, researchers have recently proposed methods that are rooted in Bayesian statistics [36] and nonlinear auto-regressive algorithm [37].

The field of artificial intelligence and machine learning has recently witnessed a boom [38] and its influence can also be observed in the multi-fidelity approaches. De *et al.* [39] developed two multi-fidelity approaches by using deep neural networks. While the first framework uses transfer learning, the second framework utilizes bi-fidelity weighted learning. Meng and Karniadakis [40], on the other hand, proposed a composite neural network that is trained based on multi-fidelity data. A physics-aware component was also added to this network; although, the physics-informed component is only used for solving inverse problems. Liu and Wang [41] proposed physics constrained multi-fidelity

neural networks for solving partial differential equations.

Based on the discussion above, (at least) two salient conclusions can be drawn about the existing multi-fidelity approaches.

- First, the existing multi-fidelity approaches assume the low-fidelity solver to be computationally efficient so that one can generate sufficient low-fidelity data. This is not always true. For example, compared to wind tunnel test data, a large eddy simulation [42, 43] solver can be treated as a low-fidelity solver. However, computational cost associated with large eddy simulation is significant, even on modern computer clusters.

- Second, the physics informed multi-fidelity approaches proposed in [41] assume that the *exact* physics corresponding to the high-fidelity data is known. This is not necessarily true. There are problems where the underlying physics is unknown [1]. Also, the apparently known governing equations are often derived based on certain assumptions and hence, only reflect the true scenario in an approximate manner.

The objective of this paper is to present a multi-fidelity physics informed deep learning framework that addresses both the limitations discussed above. Unlike some of the previous studies, it is assumed that the data-generation process for the high-fidelity data is unknown. The low-fidelity model is given by ordinary/partial differential equations. The proposed model needs no low-fidelity data; instead, the initial low fidelity model is directly trained based on the (approximate) physics of the problem. This is achieved by utilizing the recently developed physics informed deep learning algorithm. [44–47]. With this setup, important physical laws such as invariance and symmetries present in the low-fidelity model will be inherently captured by the deep learning framework. Transfer learning [45] and available high-fidelity data is then used to update the trained deep learning framework. Performance of the proposed framework is illustrated on selected reliability analysis problems from the literature.

One distinct feature of the proposed framework resides in the use of transfer learning for solving multi-fidelity problems. The basic premise in transfer learning is to store knowledge gained while solving one problem and applying it to solve a different but related problem. This helps in improving the sample efficiency of the model. In the proposed method, the knowledge gained by solving the low-fidelity problem is transferred and used for solving the high-fidelity problem. This is achieved by freezing the weights and biases of the certain layers in a deep neural network.

The rest of the paper is organized as follows. Section 2 provides details on the problem to be solved. Details about the proposed approach are presented in Section 3. Numerical results showcasing the performance of the proposed approach are presented in Section 4. Finally, Section 5 provides the concluding remarks.

## 2. Problem statement

Consider $\boldsymbol{\Xi} = (\Xi_1, \Xi_2, \ldots, \Xi_N) : \Omega \to \mathbb{R}^N$ to be an $N-$dimensional stochastic vector with cumulative distribution function

$$F_{\boldsymbol{\Xi}}(\boldsymbol{\xi}) = \mathbb{P}\left(\boldsymbol{\Xi} \leq \boldsymbol{\xi}\right), \tag{1}$$

where $\boldsymbol{\xi}$ is a realization from the random vector $\boldsymbol{\Xi}$, $\mathbb{P}\left(\cdot\right)$ represents the probability measure and $\Omega$ is the input domain. In reliability analysis, one first formulates a limit-state or performance function, $\mathcal{J}\left(\boldsymbol{\xi}\right) = 0$ such that $\mathcal{J}\left(\boldsymbol{\xi}\right) < 0$ represents the failure domain $(\Omega_f)$ and $J\left(\boldsymbol{\xi}\right) \geq 0$ represents the safe domain. Mathematically, this can be represented as

$$\Omega_f \triangleq \left\{\Xi : \mathcal{J}\left(\boldsymbol{\xi}\right) < 0\right\}. \tag{2}$$

With this consideration, the probability of failure of the system can be calculated as

$$\begin{aligned} P_f = \mathbb{P}\left(\boldsymbol{\Xi} \in \Omega_f\right) &= \int_{\Omega_f} \mathrm{d}F_{\boldsymbol{\Xi}}\left(\boldsymbol{\xi}\right) \\ &= \int_{\Omega} \mathbb{I}_{\Omega_f} \mathrm{d}F_{\boldsymbol{\Xi}}\left(\boldsymbol{\xi}\right), \end{aligned} \tag{3}$$

where $\mathbb{I}_c$ is an indicator function,

$$\mathbb{I}_c\left(\boldsymbol{\xi}\right) = \begin{cases} 1 & \text{if} \quad \boldsymbol{\xi} \in c \\ 0 & \text{if} \quad \boldsymbol{\xi} \notin c \end{cases} \tag{4}$$

Although the mathematical formulation of reliability analysis discussed above is relatively simple, the difficulty arises due to the multivariate integral in Eq. (3). Almost all the time, there exists no closed-form solution for the multivariate integral and hence, one has to rely on numerical integration techniques or asymptotic approximations. A detailed account of different reliability analysis methods can be found in [48, 49].

Another important player in reliability analysis is the limit-state function $\mathcal{J}\left(\boldsymbol{\xi}\right)$. For computing the probability of failure in Eq. (3) using numerical integration, one needs to evaluate the limit-state function repeatedly; the inherent assumption being, the mathematical model/equation for the limit-state function is known. In this regards, the accuracy of the limit-state function is of utmost importance. However, for many systems in science and engineering, the governing equation

is either not available or available in an approximate sense [1]. Under such circumstances, one has no option but to rely on data collected either from the field or from laboratory experiments. Further assuming that the system under consideration is at its design phase, the option of collecting field data becomes invalid, and performing laboratory experiments is the only feasible alternative.

Consider, $\mathcal{D}_h = [\boldsymbol{\Xi}_{hx}, \boldsymbol{u}_h]$ to be the data available from laboratory experiments where

$$\boldsymbol{\Xi}_{hx} = \boldsymbol{\Xi}_h \otimes \boldsymbol{x}_h \otimes \boldsymbol{t}_h. \tag{5}$$

$\boldsymbol{\Xi}_h = \left[\boldsymbol{\xi}^{(1)}, \ldots, \boldsymbol{\xi}^{(N_h)}\right]^T$ in Eq. (5) represents sample/data of the stochastic inputs, $\boldsymbol{x}_h = [x_1, \ldots, x_{N_x}]^T$ are the spatial locations where data is available (sensor locations) and $\boldsymbol{t}_h = [t_1, \ldots, t_{N_t}]^T$ are the times at which observations are available. The operator '$\otimes$' in Eq. (5) indicates Kronecker product and $\boldsymbol{u}_h = [u_1, \ldots u_r]$, $N_r = N_h \times N_x \times N_t$ represents the responses. '$h$' in the suffix indicates that the data-collected is high-fidelity. The limit-state function $\mathcal{J}(\boldsymbol{\xi})$ is generally expressed in terms of the response variable $u$ and a threshold $u_t$

$$\mathcal{J}(\boldsymbol{\xi}) = g\left(u(\boldsymbol{\xi}, x_i, t_j)\right) - u_t. \tag{6}$$

In case the number of data-points $N_h$ is significant, it is possible to directly train a surrogate model, $\mathcal{M} : (\boldsymbol{\xi}, x, t) \rightarrow u$ and then use it to evaluate the probability of failure in Eq. (3). Popular surrogate models available in the literature includes Gaussian process [22, 23], polynomial chaos expansion [50, 51], analysis of variance decomposition [52, 53], support vector machine [54] and hybrid polynomial correlated function expansion [55, 56]. However, in reality, the number of laboratory experiments that can be performed is limited and hence, the number of data-points available is often not sufficient for training a surrogate model. To compensate for the fact that only a limited number of high-fidelity data is available, the approximate (low-fidelity) governing equation of the system is considered,

$$u_t + h\left(u, u_x, u_{xx}, \ldots; \boldsymbol{\xi}\right) = 0. \tag{7}$$

$u_x$ and $u_{xx}$ in Eq. (7) represent the first and second derivative of $u$ with respect to $x$. As already discussed in Section 1, solving Eq. (7) to generate sufficient number of low-fidelity data can also be computationally expensive.

The objective of this paper is to develop a multi-fidelity deep learning framework that can be directly trained by using the low-fidelity model in Eq. (7) (without generating data from it) and the high-fidelity data, $\mathcal{D}_h$.

## 3. Multi-fidelity physics informed deep neural network

In this section, the proposed multi-fidelity physics informed deep neural network (MF-PIDNN) is presented. However, before proceeding to the proposed framework, details on data-driven and physics-informed deep neural networks are discussed. Data-driven and physics informed deep neural networks form the backbone of the proposed multi-fidelity approach.

### *3.1. Data-driven deep neural networks*

One of the primary components of the proposed multi-fidelity approach is a deep neural network (DNN). In this work, a fully connected DNN (FC-DNN) is used and hence, the discussion is limited to FC-DNN only. Having said that, the framework presented is generic and can be used with convolutional [57] and other types of neural networks as well.

An FC-DNN with $L-$hidden layers can be represented by using a sequence of activation functions and linear transformations

$$\mathbb{N}\left(\cdot;\boldsymbol{\theta}\right) = \left(\sigma_L \circ \mathbf{W}_{L+1}\right) \circ \cdots \circ \left(\sigma_0 \circ \mathbf{W}_1\right), \tag{8}$$

where $\sigma_j : \mathbb{R} \to \mathbb{R}$ and $\mathbf{W}_{j+1}$ respectively represents the activation function and the weight matrix associated with the edges connecting the $j-$th and $(j+1)-$th layers. The biases of the neural network are absorbed into the weight matrix $\mathbf{W}_j$; the weight matrices $\{\mathbf{W}_j\}_{j=1}^{L+1}$ are the parameters of the FC-DNN and are represented using $\boldsymbol{\theta}$. '$\circ$' in Eq. (8) represents operator composition. Note that the $0-$th layer in Eq. (8) represents the input and $(L+1)-$th layer represents the output. For using the neural network in practice, the model parameters $\boldsymbol{\theta}$ needs to be estimated. In a data-driven setting, this is achieved by minimizing a loss function. For a detailed account of different loss-functions available in the literature, interested readers may refer [38, 58]. In this work, the mean-square loss function ($\mathcal{L}_d$) has been used,

$$\mathcal{L}_d = \frac{1}{N_d}\sum_{k=1}^{N_d}\left(u_k - \hat{u}_k\right)^2. \tag{9}$$

In Eq. (9), $N_d$ represents the number of data-points, $u_k$ is the observed response corresponding to the $k-$th input, $\boldsymbol{\xi}_k$ and $\hat{u}_k$ represents the neural network predicted response corresponding to $\boldsymbol{\xi}_k$,

$$\hat{u}_k = \mathbb{N}(\boldsymbol{\xi}_k;\boldsymbol{\theta}). \tag{10}$$

The primary challenge behind the application of the DNN for engineering applications is the need for data. It is a well-acknowledged fact that DNNs are data-hungry tools [36]. Unfortunately,

for the current work, the focus is on problems where one has access to very few high-fidelity data. Therefore, the direct application of data-driven DNN is unlikely to yield satisfactory results.

### *3.2. Physics-informed deep neural networks*

To address the over-reliance of data-driven DNNs on training data, physics informed deep neural networks (PI-DNN) was proposed in [44]. The basic idea is to compute the DNN parameters directly from the physics (governing ODE/PDE) of the problem. Since its inception, the PI-DNN has been used for solving a wide range of problems in science and engineering [40, 45–47].

Consider the governing (stochastic) differential equation in Eq. (7). The objective is to solve the stochastic differential equation so as to build a mapping from the input space (stochastic, spatial and temporal inputs) to the response space. In conventional data-driven DNN, this is achieved in three simple steps

- Generate training data $\mathcal{D}_h = [\boldsymbol{\Xi}_{hx}, \boldsymbol{u}_h]$, where

$$\boldsymbol{\Xi}_{hx} = \boldsymbol{\Xi}_h \otimes \boldsymbol{x}_h \otimes \boldsymbol{t}_h, \tag{11}$$

- Represent the output $u$ using DNN,

$$u = \mathbb{N}\left(\boldsymbol{\xi}, x, t; \boldsymbol{\theta}\right). \tag{12}$$

- Compute the DNN parameters $\boldsymbol{\theta}$ by minimizing the loss-function in Eq. (9),

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_d\left(\boldsymbol{\theta}\right). \tag{13}$$

In PI-DNN, the objective is to remove the data-generation step and compute the DNN parameters $\boldsymbol{\theta}$ directly from the governing differential equation in Eq. (7). Following the method presented in [46], this is achieved in four simple steps. First, similar to the data-driven case, the response $u$ is represented by using a DNN,

$$u \approx u_{NN} = \mathbb{N}\left(\boldsymbol{\xi}, x, t; \boldsymbol{\theta}\right). \tag{14}$$

Second, the neural network outputs are modified so as to automatically satisfy the initial and Dirichlet boundary conditions.

$$\hat{u}(\boldsymbol{\xi}, x, t) = u_b(x_b, t_i) + B \cdot u_{NN}(x, t, \boldsymbol{\xi}), \tag{15}$$

where the function $B$ is defined in such a way that $B = 0$ at the boundary ($x_b$) and initial ($t_i$) points. The function $u_b(x_b, t_i)$ is defined based on the initial and boundary conditions. More details on this can be found in [45, 46]. Note that $\hat{u}(\boldsymbol{\xi}, x, t)$ can also be viewed as a DNN, $\hat{\mathbb{N}}(\boldsymbol{\xi}, x, t; \boldsymbol{\theta})$.

In the third step, collocation points for the inputs, $\mathcal{D}_c = \{\boldsymbol{\xi}_k, x_k, t_k\}_{k=1}^{N_c}$ are generated by using some suitable design of experiment scheme [52, 59]. Using the collocation points, the physics-informed loss function is formulated as

$$\mathcal{L}_p(\boldsymbol{\theta}) = \frac{1}{N_c} \sum_{i=1}^{N_c} R_i^2, \tag{16}$$

where $N_c$ is the number of collocation points and $R_i$ is the residual of the governing differential equation corresponding to the $i-$th collocation point,

$$R_i = (\hat{u}_t)_i + h\left((\hat{u})_i, (\hat{u}_x)_i, (\hat{u}_{xx})_i, \ldots; \boldsymbol{\xi}_i\right). \tag{17}$$

$(\hat{u})_i$ in Eq. (17) is obtained by substituting the $i-$th collocation point into Eq. (15). $(\hat{u}_t)_i$, $(\hat{u}_x)_i$, $(\hat{u}_{xx})_i$ are obtained by using automatic differentiation (AD) [60],

$$\begin{aligned} \hat{u}_t &= \frac{\partial \hat{u}}{\partial t} = \hat{\mathbb{N}}^t(\boldsymbol{\xi}, x, t; \boldsymbol{\theta}), \\ \hat{u}_x &= \frac{\partial \hat{u}}{\partial x} = \hat{\mathbb{N}}^x(\boldsymbol{\xi}, x, t; \boldsymbol{\theta}), \\ \hat{u}_{xx} &= \frac{\partial^2 \hat{u}}{\partial x^2} = \hat{\mathbb{N}}^{xx}(\boldsymbol{\xi}, x, t; \boldsymbol{\theta}). \end{aligned} \tag{18}$$

Note that the derivatives in Eq. (18) are also DNN. Since the DNNs in Eq. (18) are obtained by differentiating Eq. (15), they have the same architecture and same parameters; the only difference is in the form of the activation function.

In the fourth and final step, the loss function in Eq. (16) is minimized to compute the parameters of the DNN,

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_p(\boldsymbol{\theta}). \tag{19}$$

For further details on PI-DNN and its application in solving reliability analysis problems, interested readers may refer [46].

PI-DNN based reliability analysis tool proposed in [46] has two major advantages. First, unlike other reliability analysis tools, including data-driven DNN, the PI-DNN based reliability analysis tool proposed in [46] needs no simulation data. This is expected to reduce the computational cost significantly. Second, PI-DNN is trained by satisfying the governing differential equation of the system. Therefore, physical properties, such as invariance and symmetries, are satisfied. However, despite these advantages, the whole idea of PI-DNN is hinged on the fact that the exact governing

differential equation for the system under consideration is available. Unfortunately, this is not necessarily true. There exists a number of scenarios in science and engineering where the governing differential equation is not known [1]. Even if the governing equation is known, it is often based on certain assumptions and approximations. In other words, the governing differential equation only represents the reality in an approximate manner. Under such circumstances, results obtained using PI-DNN are bound to be erroneous.

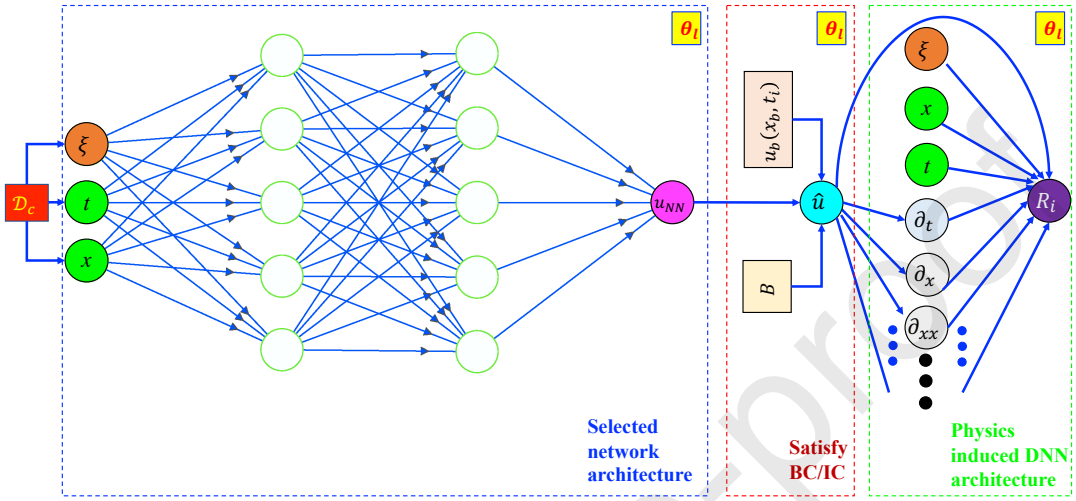
### *3.3. Proposed approach*

Neither the data-driven DNN in Subsection 3.1 nor the PI-DNN presented in Subsection 3.2 is capable of solving the reliability analysis problem defined in Section 2. The data-driven DNN fails because the number of high-fidelity data available, $N_h$ is very less. On the other hand, the PI-DNN fails as the governing differential equation in Eq. (7) only represents the actual scenario in an approximate manner. To solve the problem defined in Section 3, a multi-fidelity physics informed deep neural network (MF-PIDNN) is presented in this section. MF-PIDNN utilizes the concepts of both data-driven and physics informed DNNs. Unlike available multi-fidelity frameworks, the proposed MF-PIDNN does not assume that generating low-fidelity data is trivial. In fact, no low-fidelity data is needed for the MF-PIDNN presented here.

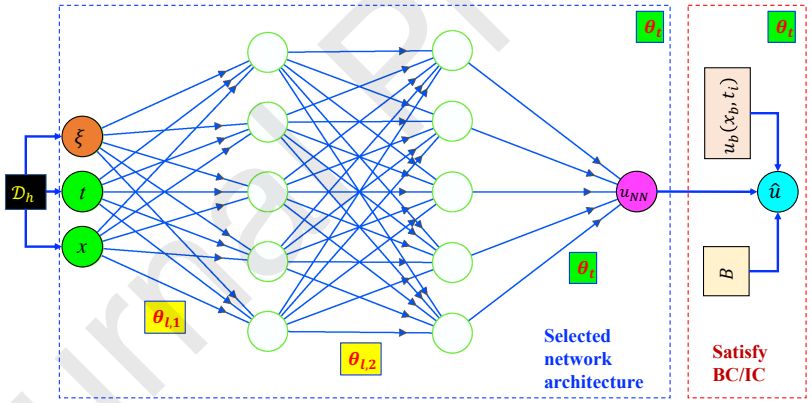
The key consideration of any multi-fidelity framework is associated with discovering and exploiting the relation between the low-fidelity and high-fidelity model/data. In most of the frameworks available in the literature, this is achieved by using two surrogates; the first surrogate is trained based on the low-fidelity data and the second surrogate is used to find the functional relation between the low-fidelity and the high-fidelity data. This paper takes a separate route; instead of using two DNNs, a single DNN is first trained for the low-fidelity model and then updated based on the high-fidelity data. For updating the DNN, the concept of transfer learning is used in this study. Note that the idea of using transfer learning in a multi-fidelity framework has previously been exploited in [39]. However, unlike the proposed framework, the algorithm presented in [39] is purely data-driven in nature.

MF-PIDNN solves the problem defined in Section 2 in two simple steps. In the first step, PI-DNN is used to solve the low-fidelity model. To that end, the exact procedure, as discussed in Subsection 3.2, is followed. In the second step, the low-fidelity PI-DNN is updated based on the high-fidelity data $\mathcal{D}_{hx}$. This is achieved by utilizing the concept of data-driven DNN. However, unlike the first step, the second step is not straight-forward. More specifically, two specific factors

are considered in this step. First, the training algorithm starts by setting the initial value of the neural network parameters to those obtained in step 1. Second, the parameters corresponding to all the layers are not updated. Instead, the concept of transfer learning [45] is used and the parameters corresponding to only the last one or two layers are updated. A schematic representation of MF-PIDNN is shown in Fig. 1. The advantage of transfer learning is three-folds.




(a) Low-fidelity training phase




(b) High-fidelity training phase

Figure 1: Schematic representation of the proposed MF-PIDNN. During the low-fidelity training phase in (a), the DNN has three building blocks. The physics induced DNN architecture is governed by the low-fidelity governing differential equation of the system. At this stage, the DNN parameters are tuned by using the collocation points $\mathcal{D}_c$ and minimizing the residual, $(R_i)$ (physics-informed loss). $\boldsymbol{\theta}_l$ (yellow box) indicates that the DNN parameters obtained at the end of the training phase. During the high-fidelity training phase in (b), the DNN parameters for all but the last one or two layers are fixed at $\boldsymbol{\theta}_l$ (yellow box)). The tunable parameters $\boldsymbol{\theta}_t$ (green box) are estimated by minimizing the mean-squared error computed using the high-fidelity data $\mathcal{D}_h$.

- First, because of transfer learning, the number of parameters to be updated is reduced. This

in turn, accelerates training of the DNN.

- Second, freezing the parameters of the initial layer ensures that the features, learned/extracted from the low-fidelity model, are retained in the network.
- Thirdly, transfer learning also ensures that the DNN does not overfit the high-fidelity data, $\mathcal{D}_h$.

The steps involved in the proposed MF-PIDNN are shown in algorithm 1. For training the MF-

**Algorithm 1:** Transfer learning based multi-fidelity physics informed deep neural network

1 **Initialize:** Provide high-fidelity data $\mathcal{D}_h$ and the low-fidelity model. Also specify the architecture of the DNN and the number of tunable layers, $l_t$ during transfer learning.

2 Express the unknown response using a DNN ; ▷ Eq. (14)

3 Modify the DNN to automatically satisfy the initial and boundary conditions ; ▷ Eq. (15)

4 Utilize the low-fidelity physics to formulate a physics-informed loss function ; ▷ Eq. (16)

5 Minimize the physics-informed loss function to compute the network parameters, $\boldsymbol{\theta}$ ; ▷ Eq. (19)

6 Freeze the DNN parameters for initial $(L - l_t + 1)$ layers.

7 Formulate data-driven loss function using $\mathcal{D}_h$ ; ▷ Eq. (9)

8 Minimize the loss-function to tune the tunable parameters

$$\boldsymbol{\theta}_t = \arg\min_{\boldsymbol{\theta}_t} \mathcal{L}_d(\boldsymbol{\theta}_t),$$

where $\boldsymbol{\theta}_t$ represents the tunable parameters.

PIDNN, RMSProp optimizer [61] followed by L-BFGS algorithm is used. Xavier initialization is used for initializing the DNN parameters. Details on the parameters settings for the optimizers are provided in Section 4. Once the MF-PIDNN is trained, it is possible to predict $u$ corresponding to some unknown inputs by using Eq. (15).

## 4. Numerical illustration

In this section, two sets of examples are presented to illustrate the performance of the proposed approach. In the first set, two examples involving function approximations are presented. The purpose of this set is to illustrate the performance of transfer learning in solving multi-fidelity

problems. Note that for these two examples, no physics-informed part is present; instead, both the low-fidelity and high-fidelity models are based on data. In the second set, four numerical examples are presented to illustrate the performance of the proposed MF-PIDNN. A wide variety of examples involving single and multiple stochastic variables, linear and non-linear problems, ordinary and partial differential equations are selected. For illustrating the performance of the proposed approach, benchmark results using Monte Carlo simulation (MCS) [62] are generated. The software accompanying the proposed approach is developed using `TenserFlow` [63]. Benchmark results for the example in subsubsection 4.2.2 are generated using the `FeNICS` package [64]. For other examples, the benchmark results are generated using `MATLAB` [65].

### *4.1. Problem set I*

In this section, two benchmark examples from the literature are presented to illustrate the performance of transfer learning in solving multi-fidelity problems. For the first example, there exists a linear correlation between the low-fidelity and the high-fidelity models. For the second example, the correlation is nonlinear. Results obtained using deep neural network, trained with only high-fidelity data, is presented for comparison. For example 1, a case study by varying the number of transfer learning layers is also presented.

#### *4.1.1. Linear correlation example*

As the first example, a continuous function with a linear correlation between the low-fidelity and high-fidelity models is considered. The low-fidelity and high-fidelity functions are represented as

$$u_L(x) = A(6x-2)^2 \sin(12x-4) + B(x-0.5) + C, \tag{20a}$$

$$u_H(x) = (6x-2)^2 \sin(12x-4). \tag{20b}$$

Clearly, there low-fidelity model in Eq. (20a) is linearly correlated with the high-fidelity model is Eq. (20b). For this example, $A = 0.5$, $B = 10$ and $C = -5$ is considered.

For solving this problem, a deep neural network having seven hidden layers is considered. The input to the network is $x$ and the output is $u$. Each of the seven hidden layers have 50 neuron each. Rectified linear unit (`ReLU`) activation is used for all but the last layer. For the last layer, linear activation function is used. For training the low-fidelity model, the RMSProp optimizer is run for 10,000 iterations. During updating the model using transfer learning, the RMSProp optimizer is

run for 1,000 iterations. The maximum number of iterations for L-BFGS optimizer is set to be 10,000.

Fig. 2 shows the results obtained using transfer learning. The number of layers allowed to be updated in transfer learning is varied from $1-4$. Result corresponding to deep neural network (same architecture and setup) trained with only high-fidelity data is also reported. Results obtained using transfer learning is found to be significantly better as compared to the results obtained using only high-fidelity data. As for the effect of the number of layers allowed to be updated, it is observed that transfer learning for this example yields the best result when the last three or four layers are allowed to update.

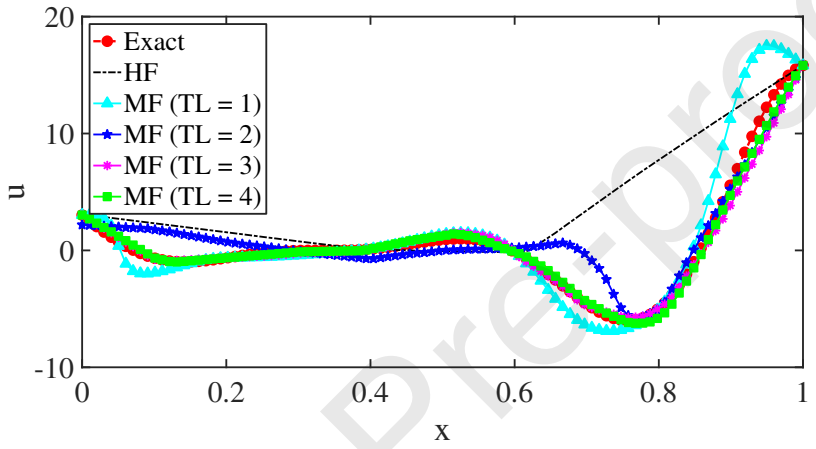



Figure 2: Performance of transfer learning is approximating a continuous function with linear correlation. The models are trained with 11 uniformly distributed low-fidelity data and 5 high-fidelity data, $x = [0.0, 0.4, 0.5, 0.6, 1.0]$.

*4.1.2. Nonlinear correlation*

In this section, a continuous function with nonlinear correlation is considered. The low-fidelity and the high-fidelity functions for this example are given as

$$u_L(x) = \sin(8\pi x), \tag{21a}$$

$$u_H(x) = (x - \sqrt{2})u_L^2(x), \tag{21b}$$

where $x \in [0, 1]$. For this example, 16 uniformly distributed samples for the high-fidelity model and 51 uniformly distributed samples for the low-fidelity model is considered.

For solving this problem, a deep neural network with seven hidden layers is considered. Each of the seven hidden layers has 100 neurons. Similar to the previous example, for all but the last layer,

`ReLU` activation function is considered. For the last layer, linear activation function is considered. The optimizer setup for this problem is kept the same as the previous problem. In the transfer learning step, weights and biases corresponding to the last two layers are allowed to be updated.

Fig. 3 shows the results obtained using the transfer learning-based approach. For the sake of comparison, the exact high-fidelity function and results obtained using deep learning with only high-fidelity data are also reported. It is observed that the results obtained using the transfer learning-based approach closely follows the pattern of the exact function. The deep neural network trained only using high-fidelity data; on the other hand, yields erroneous results. This clearly indicates the capability of the proposed transfer learning in learning from multi-fidelity data.

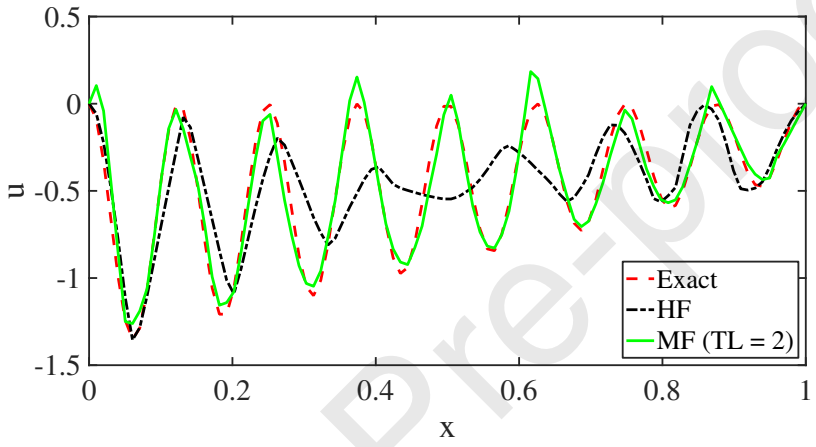



Figure 3: Performance of transfer learning in approximating continuous function with nonlinear correlation.

### *4.2. Problem set II*

In the second set, four numerical examples are presented to illustrate the performance of the proposed MF-PIDNN. The examples selected involve problems having single and multiple stochastic variables, linear and non-linear problems, ordinary and partial differential equations. Benchmark results are generated by using MCS. Comparison with low-fidelity PI-DNN and high-fidelity DNN are also presented.

#### *4.2.1. An ordinary differential equation*

As the first example, a benchmark stochastic ordinary differential equation previously studied in [66] is considered. The low-fidelity model for this problem is given by the following stochastic ordinary differential equation,

$$\frac{\mathrm{d}u_l}{\mathrm{d}t} = -Zu_l, \tag{22}$$

where $Z$ is the stochastic variable. The differential equation in Eq. (22) is subjected to the following initial condition,

$$u_l\left(t=0\right)=1.0. \tag{23}$$

The high-fidelity model, on the other hand, is represented as

$$u_h = t\sin\left(t\right)\left[\log\left(u_l^4\right)\right]^2 + 15t^3 + 1.0 \tag{24}$$

Clearly, the relation between the high-fidelity $u_h$ and the low-fidelity $u_l$ is non-linear. The limit-state function for this problem is defined as

$$\mathcal{J}(Z,t_t) = u_h(Z,t_t) - u_0, \tag{25}$$

where $u_0$ is the threshold, $u_h(Z,t_t)$ is the response and $t_t$ is the time at which the probability of failure is to be estimated. For this example, $t_t = 1.0$ and $u_0 = 18.0$ is considered. It is assumed that 15 samples from the high-fidelity model is available, and for each of the 15 high-fidelity samples, the observations are available at $t = [0.0, 1.0]$. Note that the data-generation process, i.e., Eq. (24) is not known. MF-PIDNN only have access to the high-fidelity data and the low-fidelity model in Eq. (22). For this particular problem, the stochastic variable $Z \sim \mathcal{N}(\mu, \sigma^2)$ is considered to follow Gaussian distribution with mean $\mu = -2.0$ and standard deviation $\sigma = 1.0$. MCS with $10^6$ simulations yields a probability of failure of 0.045.

For solving the problem using the proposed approach, the unknown response $u$ is first represented by using an FC-DNN with 2 inputs, 5 hidden layers and 50 neurons per hidden layer. The 2 inputs to the DNN are time $t$ and decay parameter $Z$. Hyperbolic tangent (`tanh`) activation function is considered for all but the last layer. For the last layer, linear activation function is considered. The initial conditions in Eq. (23) is automatically satisfied by modifying the DNN output, $u_{NN}$ using Eq. (15), where $u_b = 1.0$ and $B = t$,

$$\hat{u} = t \cdot u_N N + 1.0. \tag{26}$$

The residual for training the low-fidelity DNN is

$$R_i = \frac{\mathrm{d}\hat{u}_i}{\mathrm{d}t} + Z_i\hat{u}_i, \tag{27}$$

where $R_i$ is the residual and $\hat{u}_i$ is obtained from Eq. (26). '$i$' in the suffix indicates the $i-$th collocation point. For training the low-fidelity model, 8000 collocation points is used and the RMSprop optimizer is run for $15,000$ iterations. A learning rate of 0.001 is used. The other parameters of RMSprop are kept at there default values. The maximum allowable iterations for

L-BFGS optimizer is set to be $10,000$.

After training the physics-informed low-fidelity DNN, the next step is to update the model based on the high-fidelity data by using the transfer learning. The parameters corresponding to the last two layers are only updated; parameters corresponding to all other layers are kept fixed. The RMSprop optimizer is run for $10,000$ iterations and maximum allowable iterations for the L-BFGS optimizer is set to be $10,000$. For RMSProp optimizer, a learning rate of 0.001 is used.

Table 1 shows the results obtained using MCS and MF-PIDNN. Along with the probability of failure $P_f$, the reliability index $\beta$ for this problem is also reported.

$$\beta = \Phi^{-1}\left(1 - P_f\right), \tag{28}$$

where $\Phi\left(\cdot\right)$ represents cumulative distribution function of standard Gaussian distribution. The results obtained using MF-PIDNN matches exactly with the MCS results. To show the utility of the proposed approach, results obtained using only the low-fidelity PI-DNN and the high-fidelity DNN (HF-DNN) are also presented. Both low-fidelity PI-DNN (LF-PIDNN) and high-fidelity DNN are found to yield erroneous results.

Table 1: Reliability analysis results for example 1.

| **Methods** | $P_f$ | $\beta$ | $N_h$ | $N_r$ | $\epsilon = \frac{\lvert\beta_e - \beta\rvert}{\beta_e} \times 100$ |
|---|---|---|---|---|---|
| MCS | 0.045 | 1.6954 | $10^6$ | $10^6 \times 1001$ | – |
| LF-PIDNN | 0.8133 | -0.8901 | 0 | 0 | 152.5% |
| HF-DNN | 0.0 | $\infty$ | 15 | $30(15 \times 2)$ | $\infty$ |
| MF-PIDNN | 0.045 | 1.6954 | 15 | $30(15 \times 2)$ | 0.0% |

To further illustrate the performance of the MF-PIDNN, two additional case studies are performed. In the first case study, the performance of the MF-PIDNN in predicting future reliability is investigated. To that end, it is assumed that for each of the 15 high-fidelity samples, observations are available $t = [0.0, 0.5, 0.9]$, and the objective is to compute the reliability of the system at $t = 1.0$. The difficulty, in this case, arises from the fact that this is an extrapolation problem as no observation is available at or beyond $t = 1.0$. The network architecture and other parameters of MF-PIDNN are considered to be same as before; the only difference resides in the fact that the RMSProp optimizer is run for $15,000$ iterations (while updating the network using transfer learning). The results obtained are shown in Table 2. Compared to the results presented in Table 1, slight deterioration in the results have been observed; this is expected because this is an extrapolation

problem. Nonetheless, the results obtained are still significantly more accurate as compared to HF-DNN and LF-PIDNN.

Table 2: Reliability analysis results for example 1. The results presented illustrate the extrapolation capability of the MF-PIDNN.

| **Methods** | $P_f$ | $\beta$ | $N_h$ | $N_r$ | $\epsilon = \frac{\lvert\beta_e - \beta\rvert}{\beta_e} \times 100$ |
|---|---|---|---|---|---|
| MCS | 0.045 | 1.6954 | $10^6$ | $10^6 \times 1001$ | – |
| LF-PIDNN | 0.8133 | -0.8901 | 0 | 0 | 152.5% |
| HF-DNN | 0.014 | 2.9173 | 15 | $45(15 \times 3)$ | 29.60% |
| MF-PIDNN | 0.05 | 1.6449 | 15 | $45(15 \times 3)$ | 2.98% |

Finally, the performance of the MF-PIDNN with variation in the number of high-fidelity data point, $N_h$ is investigated. For each realization of $Z$, the responses are observed at $t = [0.0, 1.0]$ and the probability of failure at $t_t = 1.0$ is computed. The variation of the MF-PIDNN predicted probability of failure is shown in Fig. 4. The benchmark result obtained using MCS is also reported. With an increase in $N_h$, the MF-PIDNN predicted probability of failure converges to the MCS solution. The HF-DNN results, up to $N_h = 20$, yields erroneous results (not shown in Fig. 4). This is because, with only observations at two time-instants, the DNN fails to predict the trend of the limit-state function. MF-PIDNN, on the other hand, learns the trend from the physics of the problem and then update itself based on the high-fidelity data.

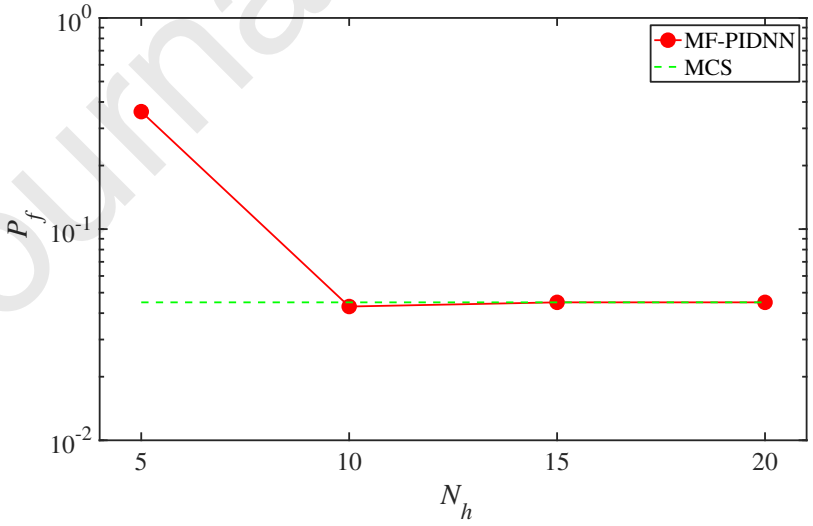



Figure 4: Variation in the MF-PIDNN predicted results with increase in number of high-fidelity data points.

*4.2.2. Burger's equation*

As the second problem, the well-known Burger's equation is considered. The high-fidelity model for this problem is

$$(u_h)_t + u_h(u_h)_x = \nu(u_h)_{xx}, \tag{29}$$

with $x \in [-1, 1]$ and $t \in [0, 12]$. $\nu > 0$ in Eq. (29) represents the viscosity. The boundary and the initial conditions for this problem are

$$u_h(t, x = -1) = 1 + \delta \quad u_h(t, x = 1) = -1, \tag{30a}$$

$$u_h(t = 0, x) = -1 + (1 + x)\left(1 + \frac{\delta}{2}\right). \tag{30b}$$

$\delta$ in Eq. (30) is a small perturbation that is applied to the left boundary. The problem as defined has a transition layer at $z$, so that $u_h(z) = 0$. As illustrated in previous studies [67, 68], the transition layer is super sensitive to $\delta$. Details on different aspects of this problem can be found in [67, 68].

The low-fidelity model, on the other hand, is considered to be

$$(u_l)_t = \nu(u_l)_{xx}. \tag{31}$$

Eq. (31) is obtained by ignoring the nonlinear term in the high-fidelity model. The initial and the boundary conditions are considered to be same as the high-fidelity model. The boundary perturbation $\delta \sim \mathcal{U}(0.0, 0.1)$ is uniformly distributed between 0.0 and 0.1. It is considered that the high-fidelity model in Eq. (24) is not known; instead, data corresponding to five realizations of $\delta$ is available. For each $\delta$, observations at 3 spatial location and 8 temporal locations are available.

$$\boldsymbol{\Xi}_{hx} = [0, 0.025, 0.05, 0.075, 0.1] \otimes [-1, 0, 1] \otimes [1, 2.14, 3.29, 4.43, 5.57, 6.71, 7.86, 9]. \tag{32}$$

The limit-state function for this problem is represented as

$$J(\delta, t_t) = -z(\delta, t) + z_0, \tag{33}$$

where $z$ represents the transition layer, $t_t$ is the time at which the reliability is to be computed and $z_0$ is the threshold. For this example, $z_0 = 0.40$ is considered. The objective is to compute the probability of failure at $t_t = 10$. Note that solution of this problem involves extrapolation as no observation at $t = 10$ or beyond is available.

For solving the problem using the proposed MF-PIDNN, $u$ is first represented by using a FC-DNN with 6 hidden layers. Each of the 6 hidden layers has 50 neurons. The DNN has 3 inputs, $x$, $t$ and $\delta$ and one output $u_{NN}$. `tanh` activation function is considered for all but the last layer. For

the last layer, linear activation function is used. To automatically satisfy the boundary and initial conditions, the DNN output is modified as

$$\hat{u} = u_h(t = 0, x) + t(1 - x)(1 + x)u_{NN}, \tag{34}$$

where $u_h(t = 0, x)$ is obtained from Eq. (30b). Using $\hat{u}$ and its derivatives, the residual of the low-fidelity model is formulated as

$$R_i = (\hat{u}_t)_i - \nu(\hat{u})_{xx})_i, \tag{35}$$

where $R_i$ is the residual. $i$ in the suffix indicates that the quantities are evaluated corresponding to the $i-$th collocation point. The physics-informed loss-function for training the low-fidelity model is formulated by using $30,000$ collocation points and Eq. (35). Because of the simplicity of the low-fidelity model, the RMSProp optimizer is run for 500 iterations, and the maximum allowable iterations for the L-BFGS optimizer is set to be 1000. The learning rate in RMSProp optimizer is set to be 0.001. Once the low-fidelity physics informed DNN is trained, the next step is to update the DNN model by using transfer learning. To retain information gained from the low-fidelity model and avoid over-fitting, parameters corresponding to only the last two layers of the DNN are allowed to update; all the other parameters are frozen. The RMSprop optimizer is run for 6000 iterations with a learning rate of 0.003. The maximum allowed iterations for the L-BFGS algorithm is set to 10,000. The L-BFGS optimizer is only allowed to update the DNN parameters corresponding to the last layer. For this problem, the MF-PIDNN is found to be highly sensitive to the initial point of the parameters and varies from run to run. Therefore, the MF-PIDNN results presented are mean predictions after running the model for 20 times.

For the purpose of validation, benchmark results using MCS with $10^4$ simulations are generated. To that end, finite element package `FeNICS` [64] is used. The same-solver is used for generating the high-fidelity data as well.

The reliability analysis results are shown in Table 3. Along with MCS and MF-PIDNN results, LF-PIDNN and HF-DNN predicted results are also presented. Similar to the previous example, both probability of failure and reliability index are reported. It is observed that MF-PIDNN predicted results are extremely close to the MCS results. HF-DNN and LF-PIDNN, on the other hand, yields erroneous results. Fig. 5 shows the performance of MF-PIDNN with increase in $N_t$ (i.e, number of time-steps at which high-fidelity data is available). It is observed that with an increase in the $N_t$, the MF-PIDNN predicted result moves closer to the MCS results. However, at $N_t = 6$ and 8,

the probability of failure obtained is found to be similar, indicating convergence of the proposed approach.

Table 3: Reliability analysis results for the Burger's equation

| **Methods** | $P_f$ | $\beta$ | $N_h$ | $N_r$ | $\epsilon = \frac{\lvert\beta-\beta_e\rvert}{\beta_e} \times 100$ |
|---|---|---|---|---|---|
| MCS | 0.2036 | 0.8288 | $10^4$ | $10^4 \times 33 \times 10^3$ | – |
| LF-PIDNN | 0 | $\infty$ | 0 | 0 | $\infty$ |
| HF-DNN | 0.932 | -1.4909 | 5 | $120(5 \times 3 \times 8)$ | 280% |
| MF-PIDNN | 0.2242 | 0.7581 | 5 | $120(5 \times 3 \times 8)$ | 8.5304% |

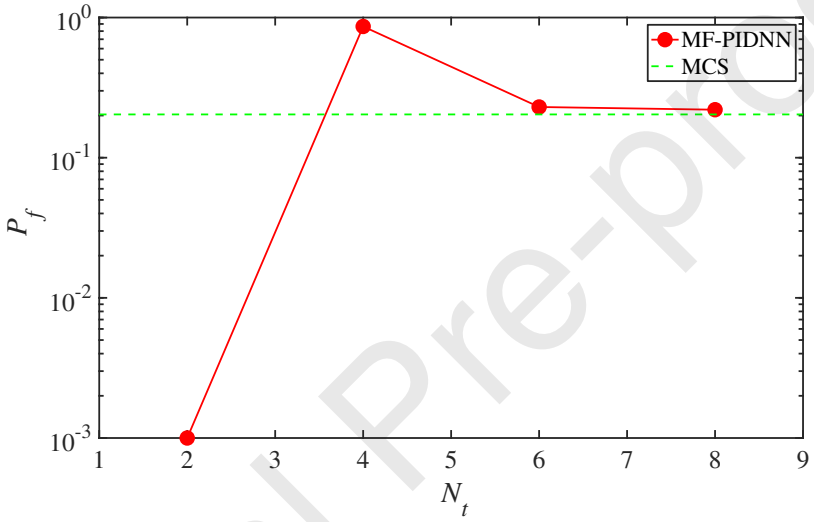



Figure 5: Variation of MF-PIDNN predicted results with $N_t$.

Finally, a case study where Eq. (29) is used as both the low-fidelity and high-fidelity model is considered. The difference between the low-fidelity and the high-fidelity model resides in the magnitude of the viscosity; for the high-fidelity model, $\nu = 10^{-3}$ and for the low-fidelity model, $\nu = 0.05$ is considered. It is considered that the high-fidelity model is not known; instead, data corresponding to six realizations of $\delta$ is available. At each $\delta$, observations at 8 uniformly distributed spatial location and 8 temporal locations are available.

$$\begin{aligned}\boldsymbol{\Xi}_{hx} = &\underbrace{[0, 0.025, 0.05, 0.075, 0.1]}_{\delta} \otimes \\ &\underbrace{[-1, -0.71, -0.43, -0.14, 0.14, 0.43, 0.71, 1]}_{x} \otimes \\ &\underbrace{[1, 2.14, 3.29, 4.43, 5.57, 6.71, 7.86, 9]}_{t}.\end{aligned} \tag{36}$$

Because of the lower value of $\nu$, a finer spatial discretization is needed for solving this problem using finite element (used for generating MCS results and HF data). Based on the convergence study, it was decided to discretize the spatial domain into 128 elements. The boundary conditions, limit-state function and other problem setup are kept the same as before.

Table 4 shows the results obtained using MCS, MF-PIDNN and HF-DNN. It is observed that the results obtained using MF-PIDNN matches closely with the MCS results. HF-DINN, on the other hand, yields erroneous results.

Table 4: Reliability analysis results for the Burger's equation (Case 2)

| **Methods** | $P_f$ | $\beta$ | $N_h$ | $N_r$ | $\epsilon = \frac{\lvert\beta-\beta_e\rvert}{\beta_e} \times 100$ |
|---|---|---|---|---|---|
| MCS | 0.2011 | 0.8377 | $10^4$ | $10^4 \times 129 \times 10^3$ | – |
| HF-DNN | 0.2741 | 0.6005 | 6 | $384(6 \times 8 \times 8)$ | 28.32% |
| MF-PIDNN | 0.2087 | 0.8109 | 6 | $384(6 \times 8 \times 8)$ | 3.19% |

*4.2.3. Nonlinear oscillator*

As the third example, a nonlinear oscillator, previously studied in [69] has been considered. The high-fidelity model for this problem is given as

$$\begin{aligned} \frac{\mathrm{d}(x_h)_1}{\mathrm{d}t} &= (x_h)_2, \\ \frac{\mathrm{d}(x_h)_2}{\mathrm{d}t} &= -\alpha_1(x_h)_2 - \alpha_2 \sin\left((x_h)_1\right), \end{aligned} \tag{37}$$

where $\alpha_1$ and $\alpha_2$ are the stochastic parameter. The initial conditions for the problem are

$$(x_h)_1(t=0) = -1.193, \quad (x_h)_2(t=0) = -3.876. \tag{38}$$

The low-fidelity model, on the other hand, is given as

$$\begin{aligned} \frac{\mathrm{d}(x_l)_1}{\mathrm{d}t} &= (x_l)_2, \\ \frac{\mathrm{d}(x_l)_2}{\mathrm{d}t} &= -\alpha_1(x_l)_2 - \alpha_2(x_h)_1. \end{aligned} \tag{39}$$

The initial condition for the low-fidelity model is considered to be same as the high-fidelity model. Similar to the previous examples, the high-fidelity equation is assumed to be unknown and one only has access to the low-fidelity model and high-fidelity data. More specifically, data corresponding to five realizations of the stochastic parameters are available. For each of the realizations, the observations are available at five equally spaced time-instants in $[0, 5]$. The realizations of the stochastic parameters are obtained using Latin hypercube sampling [70]. Following [69], the stochastic pa-

rameters $\alpha_1 \sim \mathcal{U}(0, 0.4)$ and $\alpha_2 \sim \mathcal{U}(8.8, 9.2)$ are considered to be uniformly distributed. The limit-state function for this problem is defined as

$$\mathcal{J}(\alpha_1, \alpha_2, t_t) = -|x_2(\alpha_1, \alpha_2, t_t)| + x_0, \tag{40}$$

where $x_0$ is the threshold and $t_t$ is the time at which the reliability is to be evaluated. For this problem, $t_t = 5.0$ and $x_0 = 4.0$ is considered.

To solve the problem using MF-PIDNN, $x_i$, $i = 1, 2$ is first represented using a FC-DNN having 4 hidden layers. Each hidden layer has 50 neurons. The DNN has three inputs, $t$, $\alpha_1$ and $\alpha_2$ and two outputs $x_1$ and $x_2$. `tanh` activation function is considered for all but the last layer. Linear activation function is used for the last layer. To automatically satisfy the initial conditions, the DNN output is modified as

$$\begin{aligned} \hat{x}_1 &= t \cdot x_{NN,1} - 1.193, \\ \hat{x}_2 &= t \cdot x_{NN,2} - 3.876, \end{aligned} \tag{41}$$

where $x_{NN,1}$ and $x_{NN,2}$ are the DNN outputs. Using $\hat{x}_1$ and $\hat{x}_2$, the residuals are computed,

$$\begin{aligned} R_{1,i} &= ((\hat{x}_1)_t)_i - (\hat{x}_2)_i, \\ R_{2,i} &= ((\hat{x}_2)_t)_i + (\alpha_1)_i(\hat{x}_2)_i + (\alpha_2)_i(\hat{x}_1)_i. \end{aligned} \tag{42}$$

$i$ in Eq. (42) indicates the $i-$th collocation point. Using the residuals, the physics-informed loss function for training the low-fidelity model is formulated as

$$\mathcal{L}_p(\boldsymbol{\theta}_l) = \frac{1}{N_c} \sum_{i=1}^{N_c} \left( R_{1,i}^2 + R_{2,i}^2 \right), \tag{43}$$

where $N_c$ is the number of collocation points. For this problem, 10000 collocation points have been used. The RMSProp optimizer is run for 15000 iterations with a learning-rate of 0.001. For L-BFGS optimizer, the maximum allowable iterations is set to 10000. The trained low-fidelity model is then updated by using transfer learning and high-fidelity data. Only the parameters corresponding to the last two layers of DNN are allowed to be updated. A learning rate of 0.001 is used, and the RMSProp optimizer is run for 10000 iterations. Maximum allowable iterations for the L-BFGS optimizer is set to be 10000.

The benchmark results for validation are generated by using MCS with $10^4$ simulations. To that end, the differential equations are solved using the `ODE45` routine in `MATLAB` [65]. The high-fidelity data-set discussed earlier was also generated by using the same solver.

Table 5 shows the reliability analysis results for the nonlinear oscillator problem. Similar to the previous examples, results obtained using HF-DNN and LF-PIDNN are also presented. The

MF-PIDNN is found to yield highly accurate results, matching closely with the MCS solutions. LF-PIDNN and HF-DNN yield erroneous results. The variation of probability of failure with threshold $x_0$ is shown in Fig. 6. Corresponding to all the thresholds, the MF-PIDNN predicted results matches closely with the MCS results. This indicates that the proposed MF-PIDNN is able to capture the response over the whole domain.

Table 5: Reliability analysis results for nonlinear oscillator.

| **Methods** | $P_f$ | $\beta$ | $N_h$ | $N_r$ | $\epsilon = \frac{\lvert\beta-\beta_e\rvert}{\beta_e} \times 100$ |
|---|---|---|---|---|---|
| MCS | 0.1599 | 0.9949 | 10000 | $10^4 \times 10^3$ | – |
| LF-PIDNN | 0.27 | 0.6128 | 0 | 0 | 38.41% |
| HF-DNN | 0.19 | 0.8779 | 5 | $5 \times 5$ | 11.76% |
| MF-PIDNN | 0.1576 | 1.0044 | 5 | $5 \times 5$ | 0.95% |

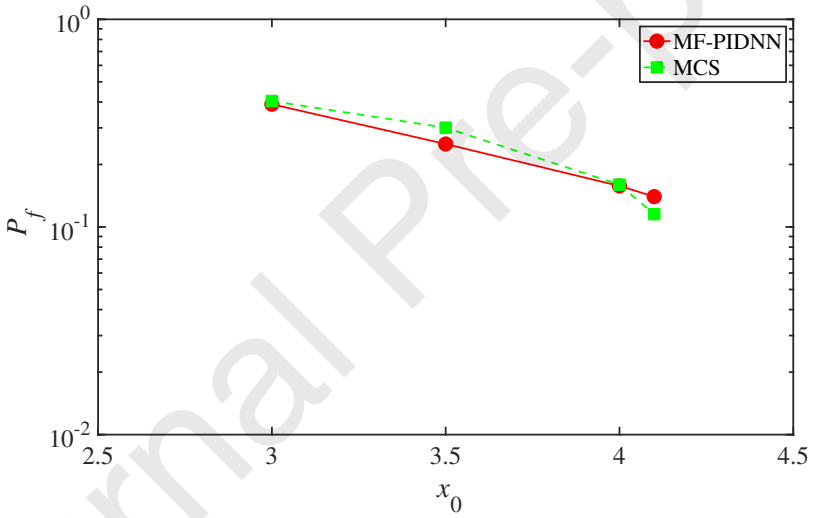



Figure 6: Variation of probability of failure with threshold $x_0$.

Lastly, to illustrate the robustness of the proposed MF-PIDNN, the same model is used to compute the probability of failure at a $t_t = 3.0$ and $x_0 = 2.0$. The results obtained are shown in Table 6. In this case also, MF-PIDNN is found to yield accurate results outperforming both HF-DNN and LF-PIDNN.

Table 6: Reliability analysis results for nonlinear oscillator at $t_t = 3.0$ and $x_0 = 2.0$

| **Methods** | $P_f$ | $\beta$ | $N_h$ | $N_r$ | $\epsilon = \frac{\lvert\beta-\beta_e\rvert}{\beta_e} \times 100$ |
|---|---|---|---|---|---|
| MCS | 0.0651 | 1.5133 | 10000 | $10^4 \times 10^3$ | – |
| LF-PIDNN | 0.98 | $-2.0537$ | 0 | 0 | 235.7% |
| HF-DNN | 0.5955 | $-0.2417$ | 5 | $5 \times 5$ | 115.97% |
| MF-PIDNN | 0.0729 | 1.4545 | 5 | $5 \times 5$ | 3.88% |

*4.2.4. Cell signaling cascade*

As the last example, a mathematical model of an autocrine cell-signaling cascade is considered.

$$\begin{aligned}
\frac{\mathrm{d}e_{1_p}}{\mathrm{d}t} &= \frac{I}{1+G_4 e_{3_p}} \frac{V_{\max,1}\left(1-e_{1_p}\right)}{K_{m,1}+\left(1-e_{1_p}\right)} - \frac{V_{\max,2} e_{1_p}}{K_{m,2}+e_{1_p}}, \\
\frac{\mathrm{d}e_{2_p}}{\mathrm{d}t} &= \frac{V_{\max,3} e_{1_p}\left(1-e_{2_p}\right)}{K_{m,3}+\left(1-e_{2_p}\right)} - \frac{V_{\max,4} e_{2_p}}{K_{m,4}+e_{2_p}}, \\
\frac{\mathrm{d}e_{3_p}}{\mathrm{d}t} &= \frac{V_{\max,5} e_{2_p}\left(1-e_{3_p}\right)}{K_{m,5}+\left(1-e_{3_p}\right)} - \frac{V_{\max,6} e_{3_p}}{K_{m,6}+e_{3_p}}, \quad t \in [0,10],
\end{aligned} \tag{44}$$

where $e_{1_p}$, $e_{2_p}$ and $e_{3_p}$ are the state variables and denotes concentrations of the active form of enzymes. $I$ in Eq. (44) is the tuning parameter. The initial conditions for this problem are

$$e_{1_p}(t=0) = 0,\ e_{2_p}(t=0) = 1.0,\ e_{3_p}(t=0) = 0. \tag{45}$$

This model was first developed in [71]. Overall the model has 13 parameters, $K_{m,1:6}$, $V_{\max,1:6}$ and $G_4$. For biological meaning and other details on the model parameters, interested readers may refer [71].

For reliability analysis, all the 13 parameters defined above are considered to be stochastic. The mean of the parameters are adopted from [71], and a 10% relative noise is added. For clarity of readers, the mean of the 13 parameters is presented in Table 7. The same parameter settings have previously been used in [69].

Table 7: Mean of the parameters for the cell signaling cascade problem

| **Parameters** | $K_{m,1:6}$ | $V_{\max,1}$ | $V_{\max,2}$ | $V_{\max,3}$ | $V_{\max,4}$ | $V_{\max,5}$ | $V_{\max,6}$ | $G_4$ |
|---|---|---|---|---|---|---|---|---|
| **Mean** | 0.2 | 0.5 | 0.15 | 0.15 | 0.15 | 0.25 | 0.05 | 2 |

A low-fidelity model for this problem is set up by considering $I = 0$. With this, the coupled differential equations in Eq. (44) is decoupled, and it becomes possible to solve the equations sequentially. Moreover, the stochastic variables $G_4$, $V_{\max,1}$ and $K_{m,1}$ become inactive. This further

complicates the problem. It is further assumed that the governing differential equation in Eq. (44) is not available; instead, responses corresponding to 10 realizations of the stochastic variables are available. For each of the 10 realizations, observations are available at 5 time-steps. The observation time-instants are equally spaced in $[4, 7]$

The limit-state function for this problem is

$$\mathcal{J}\left(\boldsymbol{\xi}\right) = e_{3_p}(\boldsymbol{\xi}, t_t) - e_{3,0}, \tag{46}$$

where $\boldsymbol{\xi} \in \mathbb{R}^{13}$ represents the stochastic variables, $e_{3,0}$ is the threshold parameter and $t_t$ is the time-instants at which the reliability is to be estimated. For this problem, $t_t = 3.0$ is considered. Since all the high-fidelity observation are available in $[4, 7]$, this is an extrapolation problem.

For reliability analysis using MF-PIDNN, the output responses are first represented by using a FC-DNN. The DNN has 14 inputs (13 stochastic variables and time), 3 outputs and 4 hidden layers. Each of the hidden layers has 100 neurons. All but the last layer of the DNN have `tanh` activation function. For the last layer, linear activation function is used. To automatically satisfy the initial conditions, the DNN outputs are modified as

$$\begin{aligned}
\hat{e}_{1_p} &= t \cdot e_{1_p,NN} \\
\hat{e}_{2_p} &= t \cdot e_{2_p,NN} + 1.0, \\
\hat{e}_{3_p} &= t \cdot e_{3_p,NN},
\end{aligned} \tag{47}$$

where $e_{1_p,NN}$, $e_{2_p,NN}$ and $e_{3_p,NN}$ are the DNN outputs. The residuals for formulating the physics-informed loss function are given as

$$\begin{aligned}
R_{1,i} &= \left((K_{m,2})_i + (\hat{e}_{1_p})_i\right)((\hat{e}_{1_p})_t)_i + (V_{\max,2})_i(\hat{e}_{1_p})_i, \\
R_{2,i} &= \left((K_{m,4})_i + (\hat{e}_{2_p})_i\right)((K_{m,3})_i + (1 - (\hat{e}_{2_p})_i)) - (V_{\max,3})_i(\hat{e}_{1_p})_i(1 - (\hat{e}_{2_p})_i), \\
R_{3,i} &= \left((K_{m,6})_i + (\hat{e}_{3_p})_i\right)((K_{m,5})_i + (1 - (\hat{e}_{3_p})_i)) - (V_{\max,5})_i(\hat{e}_{2_p})_i(1 - (\hat{e}_{3_p})_i),
\end{aligned} \tag{48}$$

where '$i$' in suffix represents the $i-$th collocation point. The residuals in Eq. (48) corresponds to the low-fidelity model and hence, $I$, $G_4$ and $V_{\max,1}$ are not present. Using the residuals, the physics-informed loss function for the low-fidelity model is computed as

$$\mathcal{L}_p(\boldsymbol{\theta}_l) = \frac{1}{N_c}\sum_{i=1}^{N_c}\sum_{k=1}^{3} R_{k,i}^2, \tag{49}$$

where $N_c$ represents the number of collocation points. For minimizing $\mathcal{L}_p(\boldsymbol{\theta})$, the RMSProp optimizer is run for 5000 iterations. A learning rate of 0.001 is used. As for the L-BFGS optimizer, the maximum allowed iterations is set to 10000. The trained low-fidelity model is then updated by

using the high-fidelity data and transfer learning. At this stage, only the parameters corresponding to the last layer is allowed to be tuned. All the other parameters are fixed at $\boldsymbol{\theta}_l$. A learning rate of 0.001 is used and the RMSProp optimizer is run for 5000 iterations. As for the L-BFGS optimizer, the maximum allowed iterations is set to be 10000.

The benchmark results for this problem are generated by using MCS with $10^4$ simulations. To that end, the `ODE45` routine available in `MATLAB` is used. The high-fidelity data discussed before were also generated by using the same procedure.

Table 8 shows the reliability analysis results for the cell signaling cascade problem. Along with MCS and MF-PIDNN, results obtained using LF-PIDNN and HF-DNN are also presented. The proposed MF-PIDNN is found to yield highly accurate results with a prediction error of 1.52%. Results obtained using LF-PIDNN and HF-DNN respectively have an error of 64.35% and 98.04%. The variation of probability of failure with the change in threshold $e_{3,0}$ is shown in Fig. 7. For all the thresholds, MF-PIDNN predicted results are found to closely match with the MCS results. This indicates that MF-PIDNN is able to capture the response over the whole domain.

Table 8: Reliability analysis results for cell signaling cascade problem

| **Methods** | $P_f$ | $\beta$ | $N_h$ | $N_r$ | $\epsilon = \frac{\lvert\beta-\beta_e\rvert}{\beta_e} \times 100$ |
|---|---|---|---|---|---|
| MCS | 0.1663 | 0.9689 | 10000 | $10^4 \times 10^3$ | – |
| LF-PIDNN | 0.3649 | 0.3454 | 0 | 0 | 64.35% |
| HF-DNN | 0.0275 | 1.9189 | 10 | $10 \times 5$ | 98.04% |
| MF-PIDNN | 0.17 | 0.9542 | 10 | $10 \times 5$ | 1.52% |

Finally, the trained MF-PIDNN is used to compute the probability of failure at different time instants. The corresponding results are illustrated in Fig. 8. To be specific, probability of failures around $t = 3, 5, 7$ and 9 are presented. The threshold $e_{3,0}$ for the four cases are set to be 0.40, 0.575, 0.70 and 0.78. MF-PIDNN for all the four cases is found to yield reasonably accurate results. Do note that high-fidelity data was only available at five equidistant time-instants between $t = 4.0$ and $t = 7.0$. The fact that the proposed approach yields reasonable results outside this domain illustrates the extrapolability of the proposed approach. This capability of the MF-PIDNN is because of the fact that some physics is learnt (and retained) from the low-fidelity data.

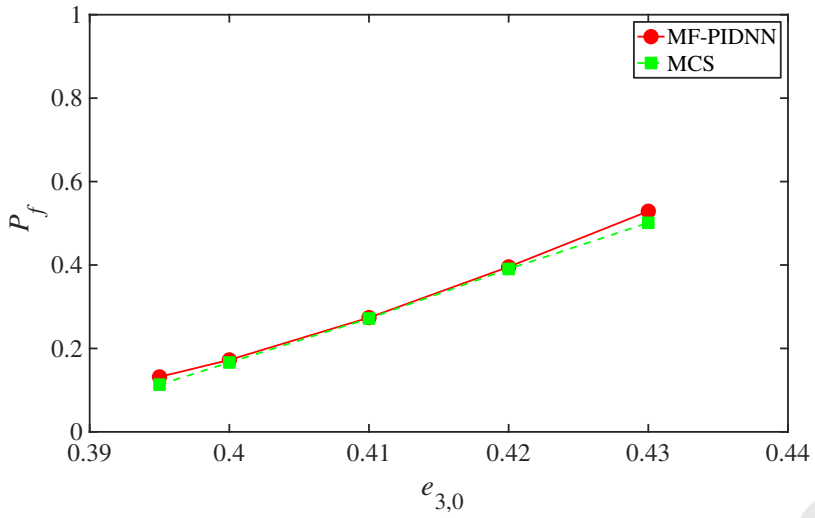



Figure 7: Variation of the probability of failure with threshold $e_{3,0}$.

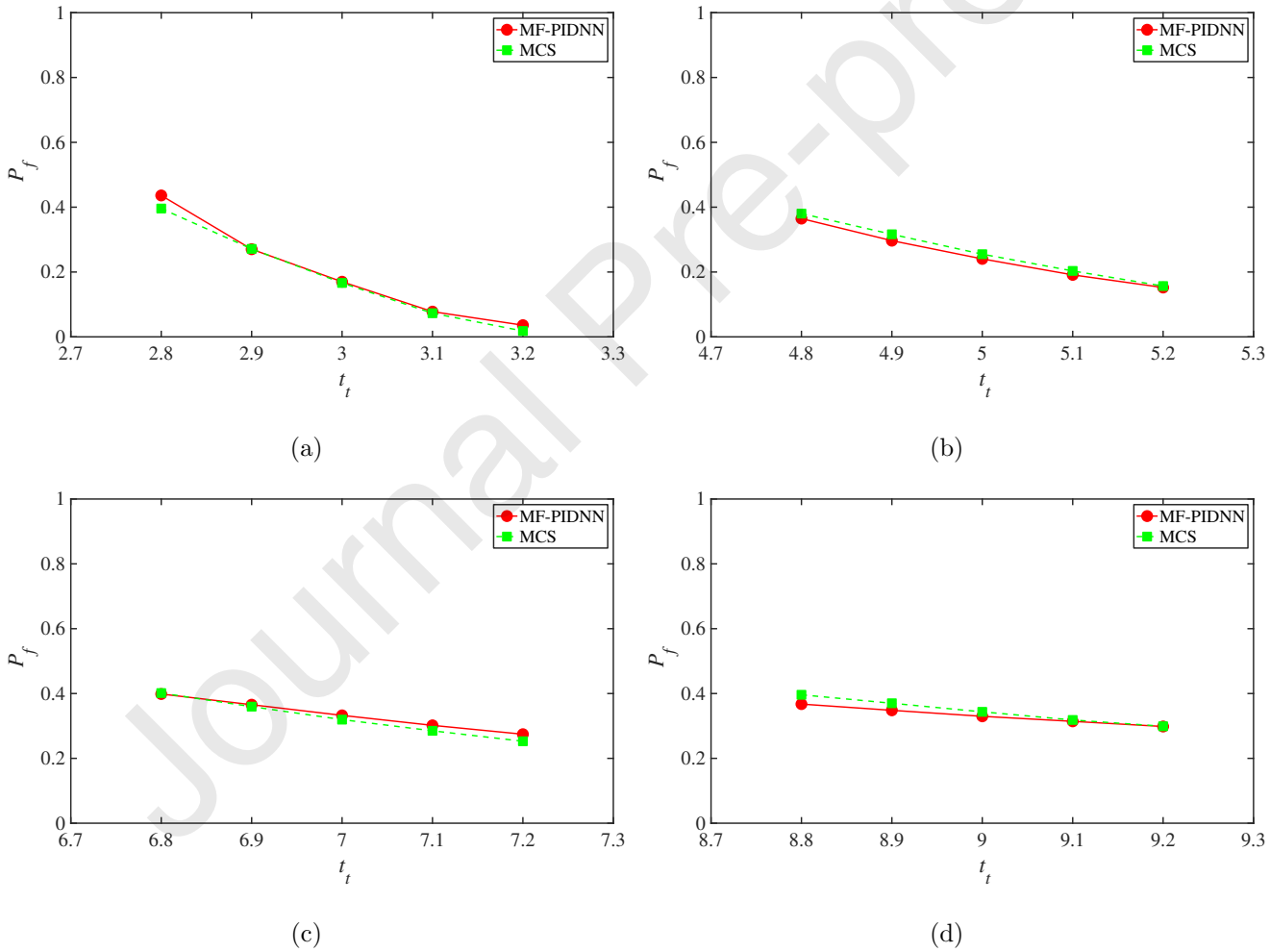



Figure 8: MF-PIDNN and MCS predicted results at different time-instants. The threshold $e_{3,0}$ for these four cases are set at (a) 0.40, (b) 0.575, (c) 0.70 and (d) 0.78.

## 5. Conclusions

In this paper, a multi-fidelity physics informed deep neural network (MF-PIDNN) is presented. The proposed approach is ideally suited for problems where the physics of the problem is known in an approximate sense (low-fidelity physics) and only a few high-fidelity data is available. MF-PIDNN blends the concepts of physics-informed and data-driven deep learning; the primary idea is to first train a low-fidelity deep learning model based on the available approximate physics and then use transfer learning to update the model based on the high-fidelity data. With this, MF-PIDNN is able to extract useful information from both the low-fidelity physics and high-fidelity data. There are two distinct advantages of MF-PIDNN. First, the low-fidelity model is directly trained from the physics of the problem and hence, no low-fidelity data is needed in this framework. Second, because of the physics-informed framework within MF-PIDNN, the proposed approach is able to capture some of the physical laws that are present in the approximate model. As a result, it provides reasonable predictions even in zones with no-data.

Two problem sets are presented to illustrate the performance of the proposed approach. In the first set, the ability of transfer learning in approximating functions from multi-fidelity data is illustrated. Functions with linear and nonlinear correlations are presented. In the second set, the proposed approach is used for solving benchmark reliability analysis problems from the literature. For all the problems, the proposed approach is able to correctly predict the probability of failure and the reliability index of the system. To illustrate the advantage of the proposed approach, results obtained are compared with those obtained from only the high-fidelity data-driven model and low-fidelity physics-driven model. The proposed approach is found to outperform both these approaches. Case studies are also presented to illustrate different features of MF-PIDNN.

Despite the several advantages of the MF-PIDNN, certain aspects can be further enhanced. For example, while updating the model using transfer learning, mean-squared loss-function with no regularization has been used. This can lead to over-fitting. One future direction is to study the effect of regularization on the results. Second, the number of tunable parameters during transfer learning are selected manually in this study. Automating the transfer learning step will be hugely beneficial. Third, the network architecture and the activation functions in this study are manually provided. Automating this will also be beneficial. In future, some of these aspects will be investigated.

**Acknowledgements**

The author would like to thank Somdatta Goswami, Tanmoy Chatterjee and Rajdip Nayek for the useful discussions during the preparation of this paper. The author also thank Govinda Anantha Padmanabha for running the MCS code for Burger's equation on his system. The `TensorFlow` codes were run on Google Colab service.

**References**

[1] S. L. Brunton, J. L. Proctor, J. N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proceedings of the national academy of sciences 113 (15) (2016) 3932–3937.

[2] D. Rodrigues, J. Belinha, F. Pires, L. Dinis, R. N. Jorge, Material homogenization technique for composites: A meshless formulation, Science and Technology of Materials 30 (1) (2018) 50–59.

[3] A. Haapio, P. Viitaniemi, Environmental effect of structural solutions and building materials to a building, Environmental impact assessment review 28 (8) (2008) 587–600.

[4] B. Mandal, A. Chakrabarti, A simple homogenization scheme for 3d finite element analysis of composite bolted joints, Composite Structures 120 (2015) 1–9.

[5] K. Ramesh, V. Ramakrishnan, Digital photoelasticity of glass: A comprehensive review, Optics and Lasers in Engineering 87 (2016) 59–74.

[6] L. Mennel, J. Symonowicz, S. Wachter, D. K. Polyushkin, A. J. Molina-Mendoza, T. Mueller, Ultrafast machine vision with 2d material neural network image sensors, Nature 579 (7797) (2020) 62–66.

[7] B. Peherstorfer, K. Willcox, M. Gunzburger, Survey of multifidelity methods in uncertainty propagation, inference, and optimization, Siam Review 60 (3) (2018) 550–591.

[8] M. Giselle Fernández-Godino, C. Park, N. H. Kim, R. T. Haftka, Issues in deciding whether to use multifidelity surrogates, AIAA Journal 57 (5) (2019) 2039–2054.

[9] S. Chakraborty, T. Chatterjee, R. Chowdhury, S. Adhikari, A surrogate based multi-fidelity approach for robust design optimization, Applied Mathematical Modelling 47 (2017) 726–744.

[10] C. Bierig, A. Chernov, Approximation of probability density functions by the multilevel monte carlo maximum entropy method, Journal of Computational Physics 314 (2016) 661–681.

[11] M. B. Giles, Multilevel monte carlo path simulation, Operations research 56 (3) (2008) 607–617.

[12] M. B. Giles, T. Nagapetyan, K. Ritter, Adaptive multilevel monte carlo approximation of distribution functions, arXiv preprint arXiv:1706.06869.

[13] S. Heinrich, Multilevel monte carlo methods, in: International Conference on Large-Scale Scientific Computing, Springer, 2001, pp. 58–67.

[14] L. Le Gratiet, J. Garnier, Recursive co-kriging model for design of computer experiments with multiple levels of fidelity, International Journal for Uncertainty Quantification 4 (5).

[15] P. Perdikaris, D. Venturi, J. O. Royset, G. E. Karniadakis, Multi-fidelity modelling via recursive co-kriging and gaussian–markov random fields, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 471 (2179) (2015) 20150018.

[16] S. Koziel, S. Ogurtsov, I. Couckuyt, T. Dhaene, Variable-fidelity electromagnetic simulations and co-kriging for accurate modeling of antennas, IEEE transactions on antennas and propagation 61 (3) (2012) 1301–1308.

[17] L. Le Gratiet, Multi-fidelity gaussian process regression for computer experiments, Ph.D. thesis, Universite Paris-Diderot (2013).

[18] S. Biswas, S. Chakraborty, S. Chandra, I. Ghosh, Kriging-based approach for estimation of vehicular speed and passenger car units on an urban arterial, Journal of Transportation Engineering, Part A: Systems 143 (3) (2017) 04016013.

[19] T. Mukhopadhyay, S. Chakraborty, S. Dey, S. Adhikari, R. Chowdhury, A critical assessment of kriging model variants for high-fidelity uncertainty quantification in dynamics of composite shells, Archives of Computational Methods in Engineering 24 (3) (2017) 495–518.

[20] A. Saha, S. Chakraborty, S. Chandra, I. Ghosh, Kriging based saturation flow models for traffic conditions in indian cities, Transportation Research Part A: Policy and Practice 118 (2018) 38–51.

[21] I. Kaymaz, Application of kriging method to structural reliability problems, Structural Safety 27 (2) (2005) 133–151.

[22] I. Bilionis, N. Zabaras, Multi-output local Gaussian process regression: Applications to uncertainty quantification, Journal of Computational Physics 231 (17) (2012) 5718–5746.

[23] I. Bilionis, N. Zabaras, B. A. Konomi, G. Lin, Multi-output separable Gaussian process: Towards an efficient, fully Bayesian paradigm for uncertainty quantification, Journal of Computational Physics 241 (2013) 212–239.

[24] R. Nayek, S. Chakraborty, S. Narasimhan, A Gaussian process latent force model for joint input-state estimation in linear structural systems, Mechanical Systems and Signal Processing 128 (2019) 497–530. doi:10.1016/j.ymssp.2019.03.048.

[25] S. Chakraborty, R. Chowdhury, Graph-Theoretic-Approach-Assisted Gaussian Process for Nonlinear Stochastic Dynamic Analysis under Generalized Loading, Journal of Engineering Mechanics 145 (12) (2019) 04019105. doi:10.1061/(ASCE)EM.1943-7889.0001685.

[26] S. Chakraborty, S. Adhikari, R. Ganguli, The role of surrogate models in the development of digital twins of dynamic systems, arXiv preprint arXiv:2001.09292.

[27] H. Babaee, P. Perdikaris, C. Chryssostomidis, G. Karniadakis, Multi-fidelity modelling of mixed convection based on experimental correlations and numerical simulations, Journal of Fluid Mechanics 809 (2016) 895–917.

[28] R. Batra, G. Pilania, B. P. Uberuaga, R. Ramprasad, Multifidelity information fusion with machine learning: A case study of dopant formation energies in hafnia, ACS applied materials & interfaces 11 (28) (2019) 24906–24918.

[29] P. Perdikaris, D. Venturi, G. E. Karniadakis, Multifidelity information fusion algorithms for high-dimensional systems and massive data sets, SIAM Journal on Scientific Computing 38 (4) (2016) B521–B538.

[30] B. Liu, S. Koziel, Q. Zhang, A multi-fidelity surrogate-model-assisted evolutionary algorithm for computationally expensive optimization problems, Journal of computational science 12 (2016) 28–37.

[31] C. Park, R. T. Haftka, N. H. Kim, Remarks on multi-fidelity surrogates, Structural and Multidisciplinary Optimization 55 (3) (2017) 1029–1050.

[32] L. Yan, T. Zhou, Adaptive multi-fidelity polynomial chaos approach to bayesian inference in inverse problems, Journal of Computational Physics 381 (2019) 110–128.

[33] P. S. Palar, T. Tsuchiya, G. T. Parks, Multi-fidelity non-intrusive polynomial chaos based on regression, Computer Methods in Applied Mechanics and Engineering 305 (2016) 579–606.

[34] H. Gao, X. Zhu, J.-X. Wang, A bi-fidelity surrogate modeling approach for uncertainty propagation in three-dimensional hemodynamic simulations, Computer Methods in Applied Mechanics and Engineering 366 (2020) 113047.

[35] A. Forrester, A. Sobester, A. Keane, Engineering design via surrogate modelling: a practical guide, John Wiley & Sons, 2008.

[36] J. Nitzler, J. Biehler, N. Fehn, P.-S. Koutsourelakis, W. A. Wall, A generalized probabilistic learning approach for multi-fidelity uncertainty propagation in complex physical simulations, arXiv preprint arXiv:2001.02892.

[37] P. Perdikaris, M. Raissi, A. Damianou, N. Lawrence, G. E. Karniadakis, Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 473 (2198) (2017) 20160751.

[38] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.

[39] S. De, J. Britton, M. Reynolds, R. Skinner, K. Jansen, A. Doostan, On transfer learning of neural networks using bi-fidelity data for uncertainty propagation, arXiv preprint arXiv:2002.04495.

[40] X. Meng, G. E. Karniadakis, A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems, Journal of Computational Physics 401 (2020) 109020.

[41] D. Liu, Y. Wang, Multi-fidelity physics-constrained neural network and its application in materials modeling, Journal of Mechanical Design 141 (12).

[42] Y. Zhiyin, Large-eddy simulation: Past, present and the future, Chinese journal of Aeronautics 28 (1) (2015) 11–24.

[43] M. F. Barone, J. Ling, K. Chowdhary, W. Davis, J. Fike, Machine learning models of errors in large eddy simulation predictions of surface pressure fluctuations, in: 47th AIAA Fluid Dynamics Conference, 2017, p. 3979.

[44] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707.

[45] S. Goswami, C. Anitescu, S. Chakraborty, T. Rabczuk, Transfer learning enhanced physics informed neural network for phase-field modeling of fracture, Theoretical and Applied Fracture Mechanics 106 (2020) 102447.

[46] S. Chakraborty, Simulation free reliability analysis: A physics-informed deep learning based approach, arXiv preprint arXiv:2005.01302.

[47] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, P. Perdikaris, Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data, Journal of Computational Physics 394 (2019) 56–81.

[48] A. Haldar, S. Mahadevan, Probability, reliability, and statistical methods in engineering design, John Wiley, 2000.

[49] A. Haldar, S. Mahadevan, Reliability assessment using stochastic finite element analysis, John Wiley & Sons, 2000.

[50] D. Xiu, G. E. Karniadakis, The Wiener-Askey polynomial chaos for stochastic differential equations, SIAM Journal on Scientific Computing 24 (2) (2002) 619–644.

[51] B. Sudret, Global sensitivity analysis using polynomial chaos expansions, Reliability Engineering & System Safety 93 (7) (2008) 964–979.

[52] S. Chakraborty, R. Chowdhury, Sequential experimental design based generalised ANOVA, Journal of Computational Physics 317 (2016) 15–32. doi:10.1016/j.jcp.2016.04.042.

[53] S. Chakraborty, R. Chowdhury, Polynomial Correlated Function Expansion, in: Modeling and Simulation Techniques in Structural Engineering, IGI Global, 2017, pp. 348–373. doi: 10.4018/978-1-5225-0588-4.ch012.

[54] A. Roy, S. Chakraborty, Support vector regression based metamodel by sequential adaptive sampling for reliability analysis of structures, Reliability Engineering & System Safety (2020) 106948.

[55] S. Chakraborty, R. Chowdhury, An efficient algorithm for building locally refined hp–adaptive h-pcfe: Application to uncertainty quantification, Journal of Computational Physics 351 (2017) 59–79.

[56] S. Chakraborty, R. Chowdhury, Hybrid framework for the estimation of rare failure event probability, Journal of Engineering Mechanics 143 (5) (2017) 04017010.

[57] Y. Zhu, N. Zabaras, Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification, Journal of Computational Physics 366 (2018) 415–447.

[58] H. D. Beale, H. B. Demuth, M. Hagan, Neural network design, Pws, Boston.

[59] B. Bhattacharyya, A Critical Appraisal of Design of Experiments for Uncertainty Quantification, Archives of Computational Methods in Engineering 25 (3) (2018) 727–751. doi: 10.1007/s11831-017-9211-x.

[60] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, Automatic differentiation in machine learning: a survey, The Journal of Machine Learning Research 18 (1) (2017) 5595–5637.

[61] T. Tieleman, G. Hinton, Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural networks for machine learning 4 (2) (2012) 26–31.

[62] R. Rubinstein, Simulation and the Monte Carlo method, Wiley, New York, U.S.A., 1981.

[63] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: A system for large-scale machine learning, in: 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), 2016, pp. 265–283.

[64] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, G. N. Wells, The fenics project version 1.5, Archive of Numerical Software 3 (100).

[65] The Mathworks Inc., Natick, Massachusetts, US., MATLAB and Statistics Toolbox Release 2019b (2019).

[66] J. Li, D. Xiu, Evaluation of failure probability via surrogate models, Journal of Computational Physics 229 (23) (2010) 8966–8980.

[67] D. Xiu, G. E. Karniadakis, Supersensitivity due to uncertain boundary conditions, International journal for numerical methods in engineering 61 (12) (2004) 2114–2138.

[68] J. Lorenz, Nonlinear singular perturbation problems and the Engquist-Osher difference scheme, Katholieke Universiteit Nijmegen. Mathematisch Instituut, 1981.

[69] T. Qin, Z. Chen, J. Jakeman, D. Xiu, A neural network approach for uncertainty quantification for time-dependent problems with random parameters, arXiv preprint arXiv:1910.07096.

[70] R. L. Iman, J. M. Davenport, D. K. Zeigler, Latin hypercube sampling (program user's guide), Tech. rep., Sandia laboratories (1980).

[71] S. Y. Shvartsman, M. P. Hagan, A. Yacoub, P. Dent, H. Wiley, D. A. Lauffenburger, Autocrine loops with positive feedback enable context-dependent cell signaling, American Journal of Physiology-Cell Physiology 282 (3) (2002) C545–C559.

**Declaration of interests**

☒ 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.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

**Author Credit Statement**

**Souvik Chakraborty:** Conceptualization, Methodology, Software/Codes**,** Data curation, Writing-Original draft preparation, Visualization, Investigation, Revision following reviewers' comments.