🟨 = Do not understand
🟩 = Review concept
🟦 = Comment / follow-up question
🟥 = Error / Inconsistency

A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems

Xuhui Meng[a], George Em Karniadakis[a,b,*]

[a] *Division of Applied Mathematics, Brown University, Providence, RI, USA, 02912*
[b] *Pacific Northwest National Laboratory, Richland, WA, USA, 99354*

Abstract

Currently the training of neural networks relies on data of comparable accuracy but in real applications only a very small set of high-fidelity data is available while inexpensive lower fidelity data may be plentiful. We propose a new composite neural network (NN) that can be trained based on multi-fidelity data. It is comprised of three NNs, with the first NN trained using the low-fidelity data and coupled to two high-fidelity NNs, one with activation functions and another one without, in order to discover and exploit nonlinear and linear correlations, respectively, between the low-fidelity and the high-fidelity data. We first demonstrate the accuracy of the new multi-fidelity NN for approximating some standard benchmark functions but also a 20-dimensional function that is not easy to approximate with other methods, e.g. Gaussian process regression. Subsequently, we extend the recently developed physics-informed neural networks (PINNs) to be trained with multi-fidelity data sets (MPINNs). MPINNs contain four fully-connected neural networks, where the first one approximates the low-fidelity data, while the second and third construct the correlation between the low- and high-fidelity data and produce the multi-fidelity approximation, which is then used in the last NN that encodes the partial differential equations (PDEs). Specifically, in the two high-fidelity NNs a relaxation parameter is introduced, which can be optimized to combine the linear and nonlinear sub-networks. By optimizing this parameter, the present model is capable of learning both the linear and complex nonlinear correlations between the low- and high-fidelity data adaptively. By training the MPINNs, we can: (1) obtain the correlation between the low- and high-fidelity data, (2) infer the quantities of interest based on a few

[*] Corresponding author
URL: george_karniadakis@brown.edu (George Em Karniadakis)

Preprint submitted to Journal Name *March 4, 2019*

scattered data, and (3) identify the unknown parameters in the PDEs. In particular, we employ the MPINNs to learn the hydraulic conductivity field for unsaturated flows as well as the reactive models for reactive transport. The results demonstrate that MPINNs can achieve relatively high accuracy based on a very small set of high-fidelity data. Despite the relatively low dimension and limited number of fidelities (two-fidelity levels) for the benchmark problems in the present study, the proposed model can be readily extended to very high-dimensional regression and classification problems involving multi-fidelity data.

Keywords: multi-fidelity, physics-informed neural networks, adversarial data, porous media, reactive transport

1. Introduction

The recent rapid developments in deep learning have also influenced the computational modeling of physical systems, e.g. in geosciences and engineering [1–4]. Generally, large numbers of high-fidelity data sets are required for optimization of complex physical systems, which may lead to computationally prohibitive costs. On the other hand, inadequate high-fidelity data result in inaccurate approximations and possibly erroneous designs. Multi-fidelity modeling has been shown to be both efficient and effective in achieving high accuracy in diverse applications by leveraging both the low- and high-fidelity data [5–8]. In the framework of multi-fidelity modeling, we assume that accurate but expensive high-fidelity data are scarce, while the cheaper and less accurate low-fidelity data are abundant. An example is the use of a few experimental measurements, which are hard to obtain, combined with synthetic data obtained from running a computational model. In many cases, the low-fidelity data can supply useful information on the trends for high-fidelity data, hence multi-fidelity modeling can greatly enhance prediction accuracy based on a small set of high-fidelity data in comparison to the single-fidelity modeling [5, 9, 10].

The construction of cross-correlation between the low- and high-fidelity data is crucial in multi-fidelity methods. Several methods have been developed to estimate such correlations, such as the response surface models [11, 12], polynomial chaos expansion [13, 14], Gaussian process regression (GPR) [6, 8, 9, 15], artificial neural networks [16], and moving least squares [17, 18]. Interested readers can refer to [19] for a comprehensive review of these methods. Among all the existing methods, the Gaussian process regression in combination with the linear autoregressive scheme has drawn much attention in a wide range of applications [8, 20]. For instance, Babaee *et al.* applied this approach for the mixed convection to propose an improved correlation for heat transfer, which outperforms existing empirical correlation [20]. We note that GPR with a linear autoregressive scheme can only capture the linear correlation between the low- and high-fidelity data. Perdikaris *et al.* then extended the method in [5] to enable it of learning complex nonlinear correlations [9]; this has been successfully employed to estimate the hydraulic conductivity based on the multi-fidelity data for pressure head in subsurface flows [21]. Although great progress has already been made, the multi-fidelity approaches based on GPR still have some limitations, e.g., approximations of discontinuous functions [7], high-dimensional problems [9], and inverse problems with strong nonlinearities (i.e., nonlinear partial differential equations) [8]. In addition, optimization for GPR is quite difficult to implement. Therefore, multi-fidelity approaches which can overcome these drawbacks are urgently needed.

Deep neural networks can easily handle problems with almost any nonlinearities at both low- and high-dimensions. In addition, the recently proposed physics-informed neural networks (PINNs) have shown *expressive power* for learning the unknown parameters or functions in inverse PDE problems with nonlinearities [22]. Examples of successful applications of PINNs include (1) learning the velocity and pressure fields based on partial observations of spatial-temporal visualizations of a passive scalar, i.e., solute concentration [23], and (2) estimation of the unknown constitutive relationship in the nonlinear diffusion equation for unsaturated flows [24]. Despite the expressive power of PINNs, it has been documented that a large set of high-fidelity data is required for identifying the unknown parameters in nonlinear PDEs. To leverage the merits of deep neural networks (DNNs) and the concept of multi-fidelity modeling, we propose to develop multi-fidelity DNNs and multi-fidelity PINNs (MPINNs), which are expected to have the following attractive features: (1) they can learn both the linear and nonlinear correlations adaptively; (2) they are suitable for high-dimensional problems; (3) they can handle inverse problems with strong nonlinearities; and (4) they are easy to implement, as we demonstrate in the present work.

The rest of the paper is organized as follows: the key concepts of multi-fidelity DNNs and MPINNs are presented in Sec. 2, while results for function approximation and inverse PDE problems are shown in Sec. 3. Finally, a summary for this work is given in Sec. 4. In the Appendix we include a basic review of the embedding theory.

2. Multi-fidelity Deep Neural Networks and MPINNs

The key starting point in multi-fidelity modeling is to discover and exploit the relation between low- and high-fidelity data [19]. A widely used comprehensive correlation [19] is expressed as

$$y_H = \rho(x) y_L + \delta(x), \tag{1}$$

where y_L and y_H are, respectively, the low- and high-fidelity data, $\rho(x)$ is the multiplicative correlation surrogate, and $\delta(x)$ is the additive correlation surrogate. It is clear that multi-fidelity models based on this relation are only capable of handling linear correlations between the two-fidelity data. However, there exist many interesting cases that go beyond the linear correlation in Eq. (1) [9]. For instance, the correlation for the low-fidelity experimental data and the high-fidelity direct numerical simulations in the mixed convection flows past a cylinder is nonlinear [9, 20]. In order to capture the nonlinear correlation, we put forth a generalized autoregressive scheme, which is expressed as

$$y_H = F(y_L) + \delta(x), \tag{2}$$

where $F(.)$ is an unknown (linear/nonlinear) function that maps the low-fidelity data to the high-fidelity level. We can further write Eq. (2) as

$$y_H = \mathcal{F}(x, y_L). \tag{3}$$

To explore the linear/nonlinear correlation adaptively, we then decompose $\mathcal{F}(.)$ into two parts, i.e., the linear and nonlinear parts, which are expressed as

$$\mathcal{F} = \mathcal{F}_l + \mathcal{F}_{nl}, \tag{4}$$

where $\mathcal{F}_l$ and $\mathcal{F}_{nl}$ denote the linear and nonlinear terms in $\mathcal{F}$, respectively. Now, we construct the correlation as

$$y_H = \alpha \mathcal{F}_l(x, y_L) + (1 - \alpha)\mathcal{F}_{nl}(x, y_L), \ \alpha \in [0, 1], \tag{5}$$

where α is hyper-parameter to be determined by the data; its value determines the degree of nonlinearity of the correlation between the high- and low-fidelity data.

The architecture of the proposed multi-fidelity DNN and MPINN is illustrated in Fig. 1, which is composed of four fully-connected neural networks. The first one $\mathcal{NN}_L(x_L, \theta)$ is employed to approximate the low-fidelity data, while the second and third NNs ($\mathcal{NN}_{H_i}(x, y_L, \beta, \gamma_i), i = 1, 2$) are for approximating the correlation for the low- and high-fidelity data; the last NN ($\mathcal{NN}_{f_e}$) is induced by encoding the governing equations, e.g. the partial differential equations (PDEs). In addition, $\mathcal{F}_l = \mathcal{NN}_{H_1}$, and $\mathcal{F}_{nl} = \mathcal{NN}_{H_2}$; θ, α, β, and $\gamma_i, i = 1, 2$ are unknown parameters of the NNs, which can be learned by minimizing the following loss function:

$$MSE = MSE_{y_L} + MSE_{y_H} + MSE_{f_e} + \lambda \sum \beta_i^2, \tag{6}$$

We do not train this network. Correct? We already know the functional form of the PDE.

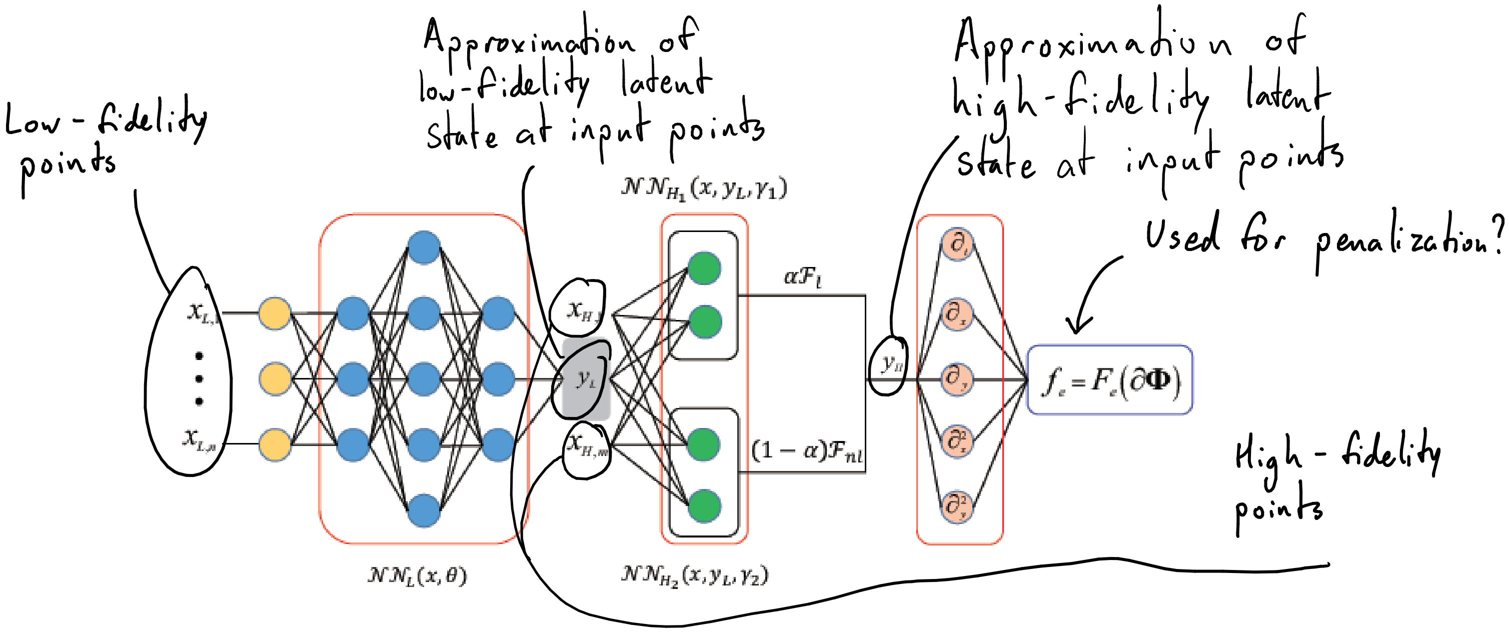

Low-fidelity points
Approximation of low-fidelity latent state at input points
Approximation of high-fidelity latent state at input points
Used for penalization?
High-fidelity points
$x_{L,1}$
$x_{L,n}$
$\mathcal{NN}_L(x,\theta)$
$x_{H,1}$
y_L
$x_{H,m}$
$\mathcal{NN}_{H_1}(x, y_L, \gamma_1)$
$\mathcal{NN}_{H_2}(x, y_L, \gamma_2)$
$\alpha\mathcal{F}_l$
$(1-\alpha)\mathcal{F}_{nl}$
y_H
∂_t
∂_x
∂_y
∂_x^2
∂_y^2
$f_e = F_e(\partial\mathbf{\Phi})$

Figure 1: Schematic of the multi-fidelity DNN and MPINN. The left box (blue nodes) represents the low-fidelity DNN $\mathcal{NN}_L(x,\theta)$ connected to the box with green dots representing two high fidelity DNNs, $\mathcal{NN}_{H_i}(x, y_L, \gamma_i)$ $(i = 1, 2)$. In the case of MPINN, the combined output of the two high-fidelity DNNs is input to an additional PDE-induced DNN. Here $\partial\mathbf{\Phi} = \left[\partial_t, \partial_x, \partial_y, \partial_x^2, \partial_y^2, ...\right] y_H$ denotes symbolically the last DNN that has a very complicated graph and its structure is determined by the specific PDE considered.

where

Why are we penalizing the gradient?

$$MSE_{y_L} = \frac{1}{N_{y_L}} \sum_{i=1}^{N_{y_L}} \left(|y_L^* - y_L|^2 + |\nabla y_L^* - \nabla y_L|^2\right), \tag{7}$$

$$MSE_{y_H} = \frac{1}{N_{y_H}} \sum_{i=1}^{N_{y_H}} \left(|y_H^* - y_H|^2\right), \tag{8}$$

$$MSE_{f_e} = \frac{1}{N_f} \sum_{i=1}^{N_f} \left(|f_e^* - f_e|^2\right). \tag{9}$$

Here, ψ ($\psi = y_L^*,\ y_H^*$, and f_e^*) denote the outputs of the $\mathcal{NN}_L$, $\mathcal{NN}_H$, and $\mathcal{NN}_{f_e}$, β is any weight in $\mathcal{NN}_L$ and $\mathcal{NN}_{H_2}$ as well as α, and λ is the L_2 regularization rate. It is worth mentioning that the boundary/initial conditions for f_e can also be added into the loss function, in a similar fashion as in the standard PINNs introduced in detail in [22] so we do not elaborate on this issue here. In the present study, the loss function is optimized using the L-BFGS method together with Xavier's initialization method, while the hyperbolic tangent function is employed as the activation function in $\mathcal{NN}_L$ and $\mathcal{NN}_{H_2}$. We note that no activation function is included in $\mathcal{NN}_{H_1}$ due to the fact that it is used to approximate the linear part of $\mathcal{F}$. Hence, the present

★ I would like to be able to play with their code. It is hard to understand how well this algorithm is doing otherwise.

★ How are the networks trained, all individually?

DNNs can dynamically learn the linear as well as the nonlinear correlations without any prior knowledge on the correlation for the low- and high-fidelity data.

3. Results and Discussion

Next we present several tests of the multi-fidelity DNN as well as the MPINN, the latter in the context of two inverse PDE problems related to geophysical applications.

3.1. Function approximation

We first demonstrate the effectiveness of this multi-fidelity modeling in approximating both continuous and discontinuous functions based on both linear and complicated nonlinear correlations between the low- and high-fidelity data.

3.1.1. Continuous function with linear correlation

We first consider a pedagogical example of approximating an one-dimensional function based on data from two levels of fidelities. The low- and high-fidelity data are generated from:

Because there is no linear part?

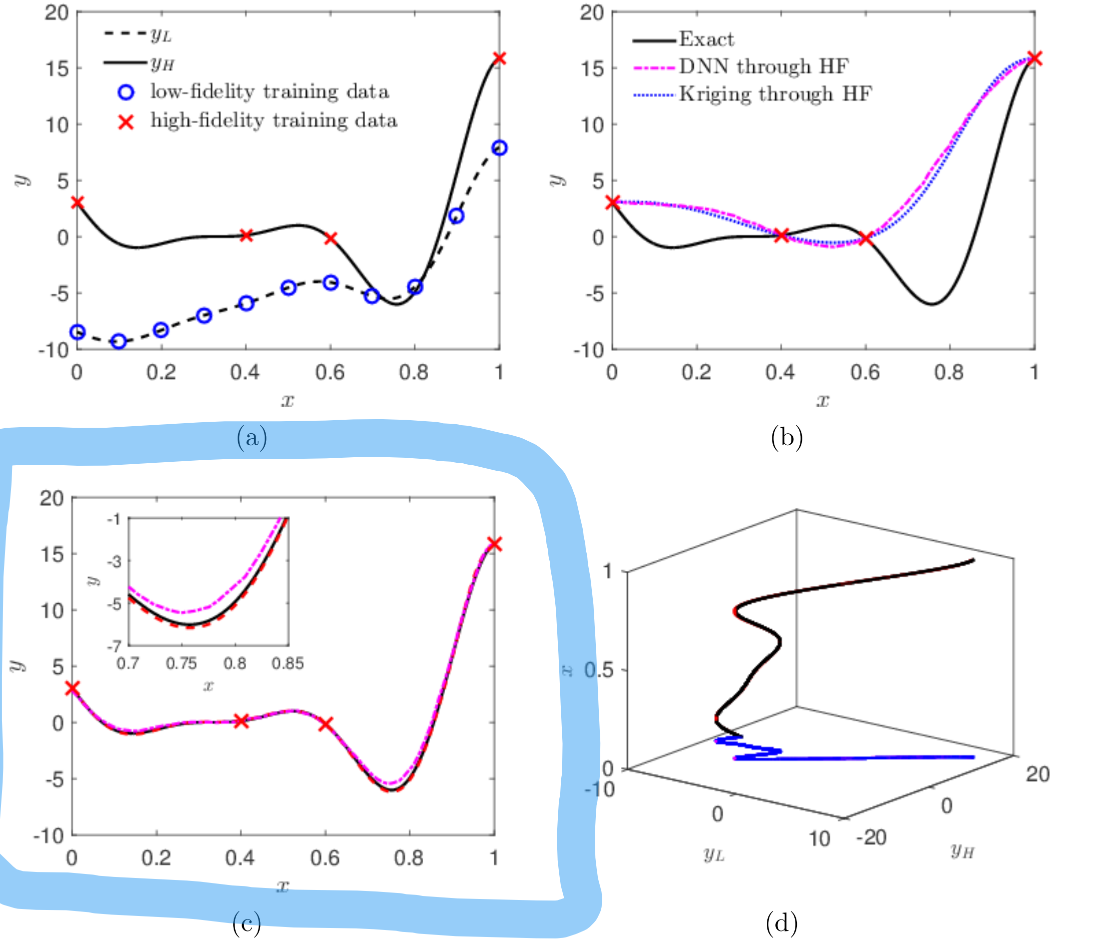
$$y_L(x) = A(6x-2)^2 \sin(12x-4) + B(x-0.5) + C, \quad x \in [0,1] \tag{10}$$

$$y_H(x) = (6x-2)^2 \sin(12x-4), \tag{11}$$

where y_H is linear with y_L, and A = 0.5, B = 10, C = -5. As shown in Fig. 2(a), the training data at the low- and high-fidelity level are $x_L = \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ and $x_H = \{0, 0.4, 0.6, 1\}$, respectively.

We first try to predict the true function using the high-fidelity data only. For this case, we only need to keep $\mathcal{NN}_{H_2}$ (Fig. 1). In addition, the input for $\mathcal{NN}_{H_2}$ becomes x because no low-fidelity data are available. Here 4 hidden layers and 20 neurons per layer are adopted in $\mathcal{NN}_{H_2}$ and no regularization is used. The learning rate is set as 0.001. As we can see in Fig. 2(b), the present model provides inaccurate predictions due to the lack of sufficient high-fidelity data. Furthermore, we also plot the predictive posterior means of the Kriging [6], which is noted to be similar as the results from the $\mathcal{NN}_{H_2}$. Keeping the high-fidelity data fixed, we try to improve the accuracy of prediction by adding low-fidelity data (Fig. 2(a)). In this case, the last DNN for the PDE is discarded. Here 2 hidden layers and 20 neurons per layer are used in $\mathcal{NN}_L$, while 2 hidden layers with 10 neurons per layer are employed for $\mathcal{NN}_{H_2}$, and only 1 hidden with 10 neurons are used in $\mathcal{NN}_{H_1}$ (The size of $\mathcal{NN}_{H_1}$ is kept identical in all of the following cases). The regularization rate is set to $\lambda = 10^{-2}$ with a learning rate 0.001. As shown in Fig. 2(c), the present model provides accurate

I want to review the code to see precisely how this model is trained.



(a) (b)

(c) (d)

Figure 2: Approximation of a continuous function from multi-fidelity data with *linear* correlation. (a) Training data at low- (11 data points) and high-fidelity levels (4 data points). (b) Predictions from DNN using high-fidelity data only; also included are the results of Kriging. (c) Predictions from the multi-fidelity DNN (Red dashed line) and Co-Kriging [6]) (Magenta dotted line). (d) The Red dashed line in the (x, y_L, y_H) plane represents Eq. (3) (on top of the exact Black solid line) and the Red dashed line in the (y_L, y_H) plane represents the correlation discovered between the high- and low-fidelity data; the Blue solid line is the exact correlation.

predictions for the high-fidelity profile. In addition, the prediction using the Co-Kriging is displayed in Fig. 2(c) [6]. We see that the learned profiles from these two methods are similar, while the result from the present model is slightly better than the Co-Kriging, which can be seen in the inset of Fig. 2(c). Finally, the estimated correlation is illustrated in Fig. 2(d), which also agrees quite well with the exact result. Unlike the Co-Kriging/GPR, no prior knowledge on the correlation between the low- and high-fidelity data is needed in the multi-fidelity DNN, indicating that the present model can learn the correlation dynamically based on the given data.

The size of the neural network (e.g., depth and width) has a strong effect on the predictive accuracy [22], which is also investigated here. Since we have sufficient low-fidelity data, it is easy to find an appropriate size for $\mathcal{NN}_L$ to approximate the low-fidelity function. Therefore, the particular focus is put on the size of $\mathcal{NN}_{H_2}$ due to the fact that the few high-fidelity data may yield overfitting. Note that since the correlation between the low- and high-fidelity data is relatively simple, there is no need to set the $\mathcal{NN}_{H_2}$ to have a large size. Hence, we limit the ranges of the depth (i.e., l) and width (i.e., w) as: $l \in [1, 4]$ and $w \in [2, 32]$, respectively. Considering that a random initialization is utilized, we perform ten runs for each case with different depth and width. The mean and standard deviation for the relative L_2 errors defined as

$$E = \frac{1}{N}\sum_{n=1}^{n=N}\sqrt{\frac{\sum_j (y_j^* - y_j)^2}{\sum y_j^2}},\ \sigma = \sqrt{\frac{\sum_{n=1}^{n=N}(E_n - E)^2}{N}}, \tag{12}$$

are used to quantify the effect of the size of the $\mathcal{NN}_{H_2}$. In Eq. (12), E is the mean relative L_2 errors, n is the index of each run, N is the total number of runs ($N = 10$), j is the index for each sample data points, E_n is the relative L_2 error for the $n - th$ run, and the definitions of y^* and y are the same as those in Sec. 1. As shown in Table 1, the computational errors for $\mathcal{NN}_{H_2}$ with different depth and width are almost the same. In addition, the standard deviation for the relative L_2 errors are not presented because they are less than 10^{-5} for each case. All these results demonstrate the robustness of the multi-fidelity DNNs. To reduce the computational cost as well and retain the accuracy, a good choice for the size of $\mathcal{NN}_{H_2}$ may be $l \in [1, 2]$ and $w \in [4, 20]$ in low dimensions.

3.1.2. Discontinuous function with linear correlation

As mentioned in [7], the approximation of a discontinuous function using GPR is challenging due to the continuous kernel employed. We then proceed to test the capability of the present model for approximating discontinuous functions. The low-

Table 1: Mean relative $L_2 (\times 10^{-3})$ for NNs with different sizes.

Width Depth	4	8	16	32
1	3.1	3.0	4.6	2.9
2	3.4	3.0	3.0	3.1
3	3.1	3.1	3.1	3.0
4	3.0	3.0	3.0	3.0

and high-fidelity data are generated by the following "Forrester" functions with jump [7]:

$$y_L(x) = \begin{cases} 0.5(6x-2)^2 \sin(12x-4) + 10(x-0.5) - 5, & 0 \le x \le 0.5, \\ 3 + 0.5(6x-2)^2 \sin(12x-4) + 10(x-0.5) - 5, & 0.5 < x \le 1, \end{cases} \tag{13}$$

and

$$y_H(x) = \begin{cases} 2y_L(x) - 20x + 20, & 0 \le x \le 0.5, \\ 4 + 2y_L(x) - 20x + 20, & 0.5 < x \le 1. \end{cases} \tag{14}$$

As illustrated in Fig. 3(a), 38 and 5 sampling data points are employed as the training data at the low- and high-fidelity level, respectively. The learning rate is again set as 0.001 for all test cases here. Similarly, we employ the $\mathcal{NN}_{H_2}$ $(l \times w = 4 \times 20)$ to predict the high-fidelity values on the basis of the given high-fidelity data only, but the corresponding prediction is not good (Fig. 3(b)). However, using the multi-fidelity data, the present model can provide quite accurate predictions for the high-fidelity profile (Fig. 3(c)). Remarkably, the multi-fidelity DNN can capture the discontinuity at $x = 0.5$ at the high-fidelity level quite well even though no data are available in the range $0.4 < x < 0.6$. This is reasonable because the low- and high-fidelity data share the same trend as $0.4 < x < 0.6$, yielding the correct predictions of the high-fidelity values in this zone. Furthermore, the learned correlation is displayed in Fig. 3(d), which shows only slight differences from the exact correlation.

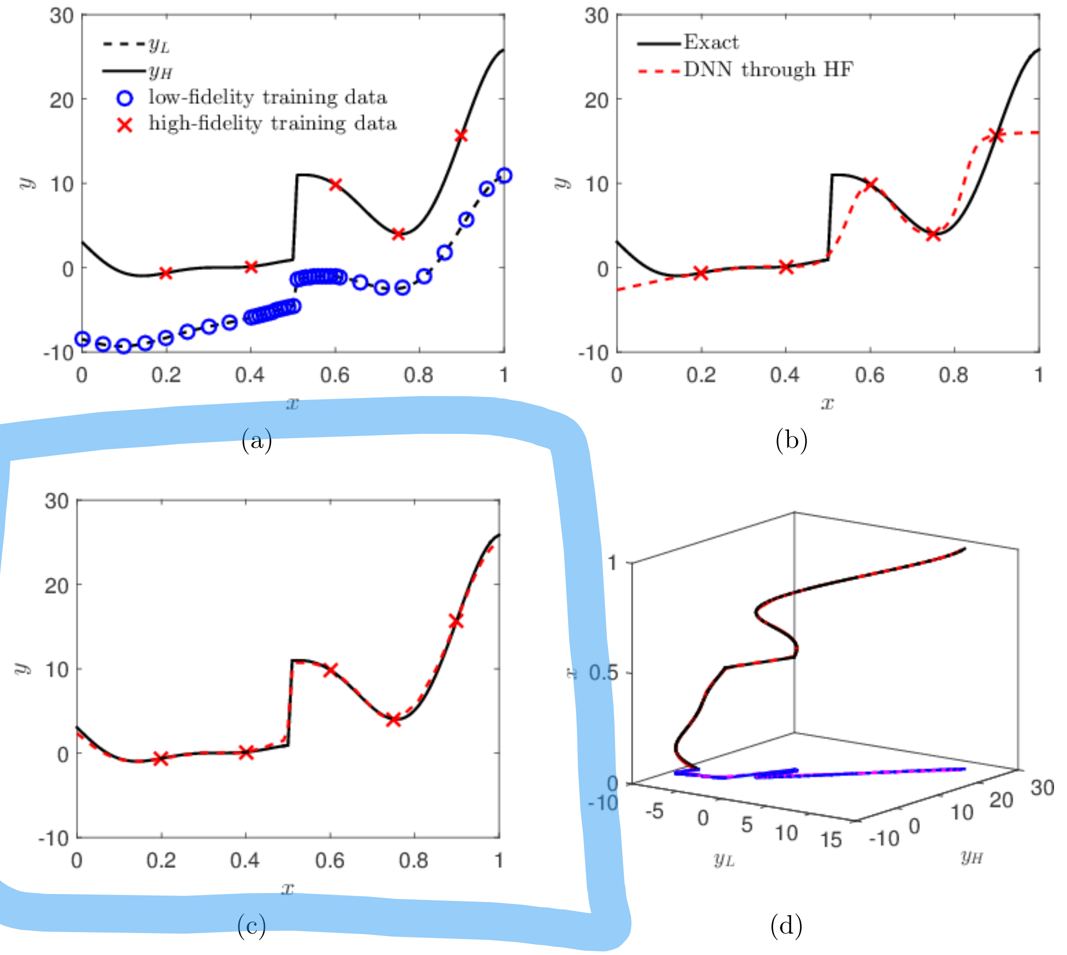
3.1.3. Continuous function with nonlinear correlation

To test the present model for capturing complicated nonlinear correlations between the low- and high-fidelity data, we further consider the following case [9]:

$$y_L(x) = \sin(8\pi x), \quad x \in [0, 1], \tag{15}$$

$$y_H(x) = (x - \sqrt{2}) y_L^2(x). \tag{16}$$

How much data was used to train this?
In particular, I want to see how correlations are trained.



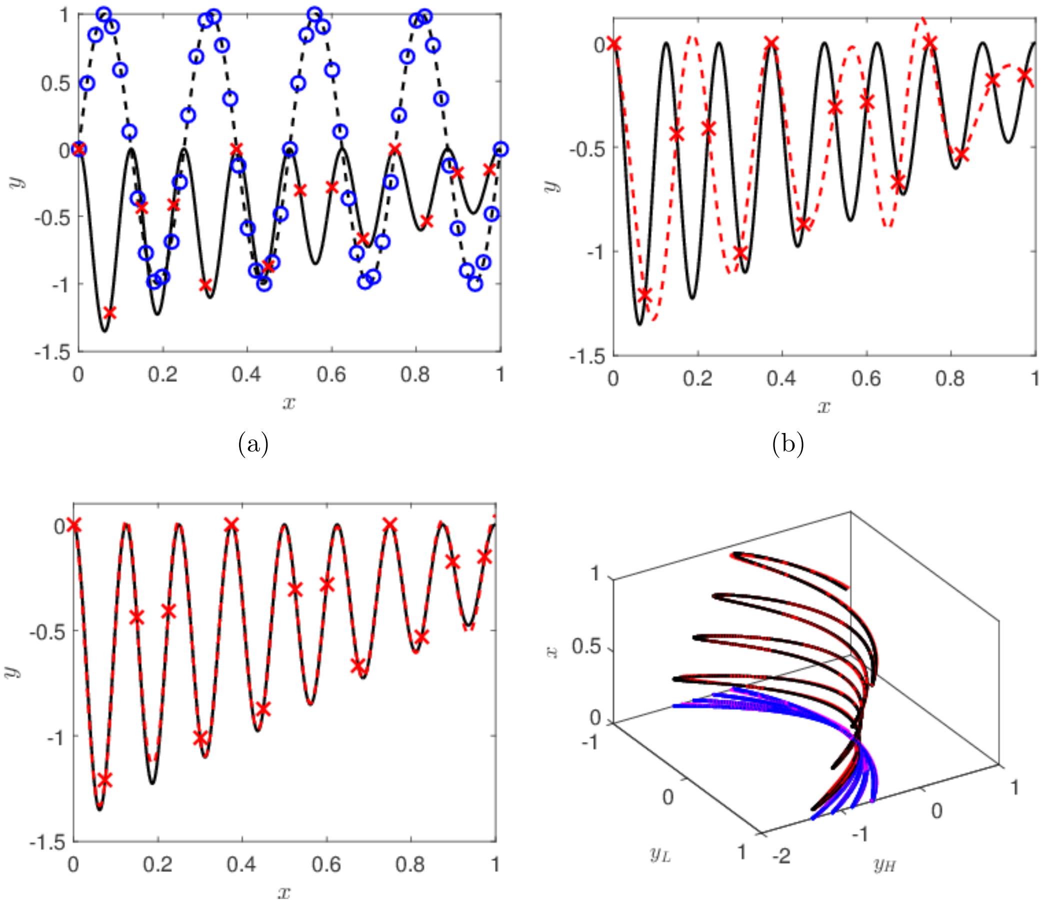
(a)

(b)

(c)

(d)

Figure 3: Approximation of a discontinuous function from multi-fidelity data with *linear* correlation. (a) Training data at low- (38 data points) and high-fidelity levels (5 data points). (b) Predictions from DNN using high-fidelity data only (Red dash line); also included is the exact curve (Black solid line). (c) Predictions from multi-fidelity DNN for high-fidelity (Red dash line). (d) The Red dashed line in the (x, y_L, y_H) plane represents Eq. (3) (on top of the exact Black solid line) and the Red dashed line in the (y_L, y_H) plane represents the correlation discovered between the high- and low-fidelity data; the Blue line is the exact correlation.



(a)

(b)

(c)

(d)

Figure 4: Approximation of a continuous function from multi-fidelity data with *nonlinear* correlation. (a) Training data at low- (51 data points) and high-fidelity levels (14 data points). Black solid line: High-fidelity values, Black dashed line: Low-fidelity values, Red cross: High-fidelity training data, Blue circle: Low-fidelity training data. (b) Predictions from high-fidelity DNN (Red dashed line); Black solid line: Exact values. (c) Predictions from multi-fidelity DNN for high-fidelity (Red dash line). (d) The Red dashed line in (x, y_L, y_H) represents Eq. (3) (on top of the exact Black solid line) and the Red dashed line in (y_L, y_H) represents the correlation discovered between the high- and low-fidelity data; the Blue line is the exact correlation.

Here, we employ 51 and 14 data points (uniformly distributed) for low- and high-fidelity, respectively, as the training data, (Fig. 4(a)). The learning rate for all test cases is still 0.001. As before, the $\mathcal{NN}_{H_2}$ ($l \times w = 4 \times 20$) cannot provide accurate predictions for the high-fidelity values using only the few high-fidelity data points as displayed in Fig. 4(b). We then test the performance of the multi-fidelity DNN based on the multi-fidelity training data. Four hidden layers and 20 neurons per layer are used in $\mathcal{NN}_L$, and 2 hidden layers with 10 neurons per layer are utilized for $\mathcal{NN}_{H_2}$. Again, the predicted profile from the present model agrees well with the exact profile at the high-fidelity level, as shown in Fig. 4(c). It is interesting to find that the multi-fidelity DNN can still provide accurate predictions for the high-fidelity profile even though the trend for the low-fidelity data is opposite to that of the high-fidelity data, e.g., $0 < x < 0.2$, a case of *adversarial* type of data. In addition, the learned correlation between the low- and high-fidelity data agree well with the exact one as illustrated in Fig. 4(d), indicating that the multi-fidelity DNN is capable of discovering the non-trivial underlying correlation on the basis of training data.

All the examples are deterministic. I would like to see how this responds to noise.

3.1.4. Phase-shifted oscillations

For more complicated correlations between the low and high-fidelity data, we can easily extend the multi-fidelity DNN based on the "embedding theory" to enhance the capability for learning more complex correlations [25] (For more details on the embedding theory, refer to Appendix A). Here, we consider the following low-/high-fidelity functions with phase errors [25]:

$$y_H(x) = x^2 + \sin^2(8\pi x + \pi/10), \tag{17}$$

$$y_L(x) = \sin(8\pi x). \tag{18}$$

We can further write y_H in terms of y_L as

$$y_H = x^2 + (y_L \cos(\pi/10) + y_L^{(1)} \sin(\pi/10))^2, \tag{19}$$

where $y_L^{(1)}$ denotes the first derivatives of y_L. The relation between the low- and high-fidelity data is displayed in Fig. 5(a), which is rather complicated. The performance of the multi-fidelity DNN for this case will be tested next. To approximate the high-fidelity function, we select 51 and 16 uniformly distributed sample points as the training data for low- and high-fidelity values, respectively (Fig. 5(b)). The selected learning rate for all test cases is 0.001. Here, we test two types of inputs for $\mathcal{NN}_{H_2}$, i.e., $[x, y_L(x)]$ (Method I), and $[x, y_L(x), y_L(x-\tau)]$ (Method II) (τ is the delay). Four hidden layers and 20 neurons per layer are used in $\mathcal{NN}_L$, and 2 hidden layers with 10 neurons per layer are utilized for $\mathcal{NN}_{H_2}$. As shown in Fig. 5, it is interesting to find that Method II provides accurate predictions for the high-fidelity values (Fig. 5(d)), while Method I fails (Fig. 5(c)). As mentioned in [25], the term $y_L(x-\tau)$ can be viewed as an implicit approximation for $y_L^{(1)}$, which enables Method II to capture the correlation in Eq. (19) based only on a small number of high-fidelity data points. However, given that no information on $y_L^{(1)}$ is available in Method I, the present datasets are insufficient to obtain the correct correlation.

3.1.5. 20-dimensional function approximation

In principle, the new multi-fidelity DNN can approximate any high-dimensional function so here we take a modest size so that is not computationally expensive to train the DNN. Specifically, we generate the low- and high-fidelity data for a 20-dimensional function from the following equations: [26]

$$y_H(x) = (x_1-1)^2 + \sum_{i=2}^{20}\left(2x_i^2 - x_{i-1}\right)^2, x_i \in [-3,3], i = 1,2,...,20, \tag{20}$$

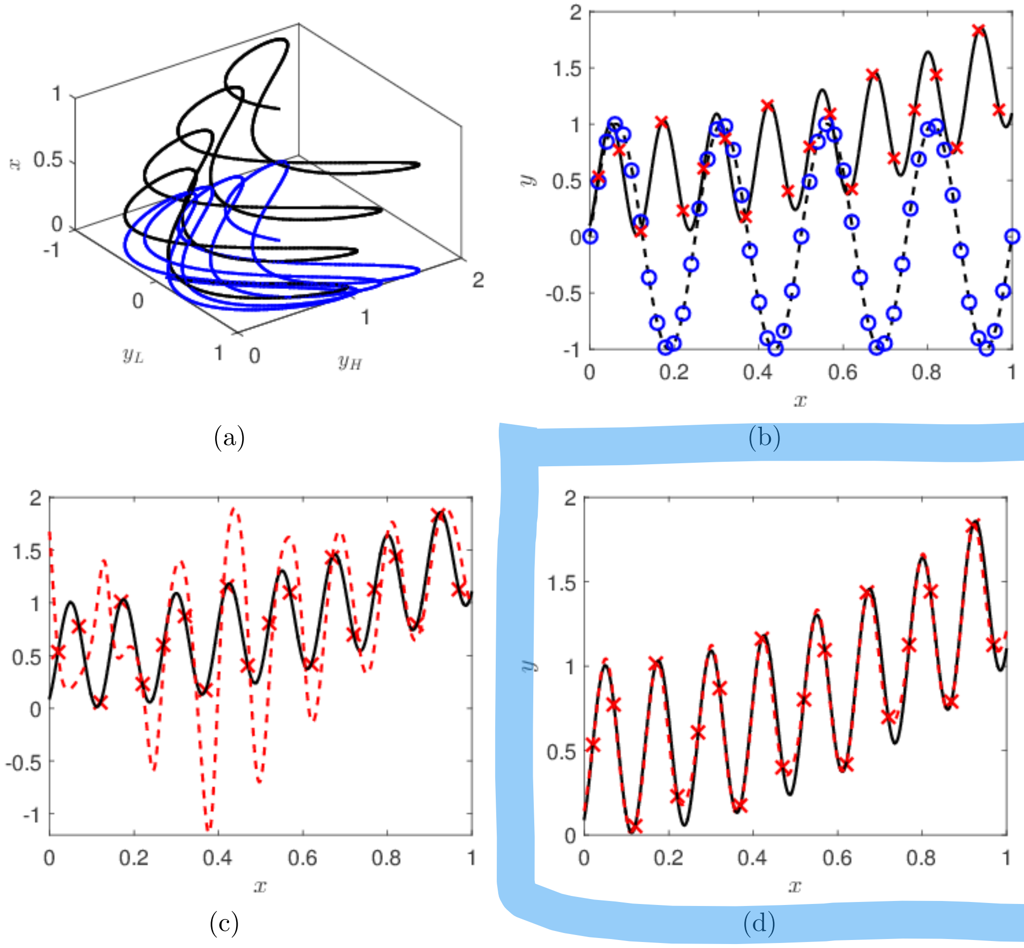
$$y_L(x) = 0.8y_H(x) - \sum_{i=1}^{19} 0.4x_i x_{i+1} - 50. \tag{21}$$

As shown in Fig. 6(a), using only the available high-fidelity data does not lead to an accurate function approximation but using the multi-fidelity DNN approach gives excellent results as shown in Fig. 6(b).

In summary, in this section we have demonstrated using different data sets and correlations that multi-fidelity DNNs can adaptively learn the underlying correlation between the low- and high-fidelity data from the given datasets without any prior assumption on the correlation. In addition, they can be applied to high-dimensional cases, hence outperforming GPR [9]. Finally, the present framework can be easily extended based on the embedding theory to non-functional correlations, which enables multi-fidelity DNNs to learn more complicated nonlinear correlations induced by phase errors of the low-fidelity data (adversarial data).

3.2. Inverse PDE problems with nonlinearities

In this section, we will apply the multi-fidelity PINNs (MPINNs) to two inverse PDE problems with nonlinearities, specifically, unsaturated flows and reactive transport in porous media. We assume that the hydraulic conductivity is first estimated based on scarce high-fidelity measurements of the pressure head. Subsequently, the reactive models are further learned given a small set of high-fidelity observations of the solute concentration.



(a)

(b)

(c)

(d)

Figure 5: Approximation of continuous function from multi-fidelity data with phase-shifted oscillations and *highly-nonlinear* correlation: (a) Correlation among x, y_L, and y_H. The Blue line represents the projection in the (y_L, y_H) plane. (b) Training data for y_L and y_H. Black solid line: Exact high-fidelity values; Black dashed line: Exact low-fidelity values; Red cross: High-fidelity training data; Blue circle: Low-fidelity training data. (c) Predictions from Method I (without time-delay) (Red dashed line). (d) Predictions from Method II (with time-delay) (Red dashed line). The learned optimal value for τ is 4.49×10^{-2}.

Why did they stop bench marking with kriging? I think this paper needs more benchmarking.

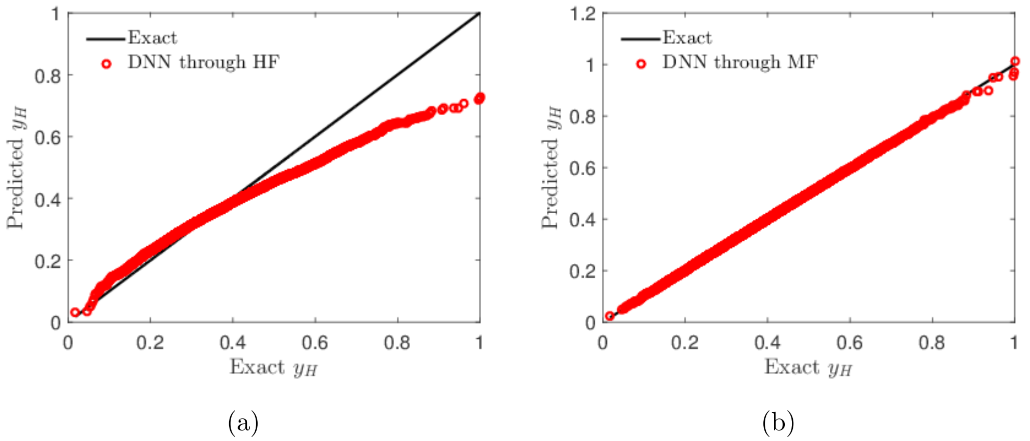


Figure 6: Approximations of the 20-dimensional function (learning rate: 0.001). (a) Single-fidelity predictions from high-fidelity data. $\mathcal{NN}_{H_2} \to 4 \times 160$ with 5000 randomly selected high-fidelity data, and 10000 test data at random locations. (b) Multi-fidelity DNN predictions. $\mathcal{NN}_L \to 4\times128$, $\mathcal{NN}_{H_2} \to 2 \times 64$ with 30000 and 5000 randomly selected low-/high-fidelity data, and 10000 test data at random locations.

3.2.1. Learning the hydraulic conductivity for nonlinear unsaturated flows

Unsaturated flows play an important role in the ground-subsurface water interaction zone [27, 28]. Here we consider a steady unsaturated flow in an one-dimensional (1D) column with a variable water content, which can be described by the following equation as

$$\partial_x \left(K(h)\partial_x h\right) = 0. \tag{22}$$

We consider two types of boundary conditions, i.e., (1) constant flux at the inlet and constant pressure head at the outlet, $q = -K\partial_x h = q_0,\ x = 0;\ h = h_1,\ x = L_x$ (Case I), and (2) constant pressure head at both the inlet and outlet, $h = h_0,\ x = 0;\ h = h_1,\ x = L_x$ (Case II). Here $L_x = 200cm$ is the length of the column, h is the pressure head, h_0 and h_1 are, respectively, the pressure head at the inlet and outlet, q represents the flux, and q_0 is the flux at the inlet, which is a constant. In addition, $K(h)$ denotes the pressure-dependent hydraulic conductivity, which is expressed as

$$K(h) = K_s S_e^{1/2}\left[1 - (1 - S_e^{1/m})^m\right]^2, \tag{23}$$

where K_s is the saturated hydraulic conductivity, and S_e is the effective saturation that is a function of h. It is noted that several models have been developed to characterize S_e but among them, the van Genuchten model is the most widely used

[29], which reads as follows:

$$S_e = \frac{1}{(1+|\alpha_0 h|^n)^m},\ m = 1 - 1/n. \tag{24}$$

In Eq. (24), α_0 is related to the inverse of the air entry suction, and m represents a measure of the pore-size distribution. To obtain the velocity field for later applications, we should first obtain the distribution of $K(h)$. Unfortunately, both parameters depend on the geometry of porous medium and are difficult to measure directly. We note that the pressure head can be measured more easily in comparison to α_0 and m. Therefore, we assume that partial measurements of h are available without the direct measurements of α_0 and m. The objective is to estimate α_0 and m based on the observations of h. Then, we can compute the distribution of $K(h)$ according to Eqs. (23) and (24).

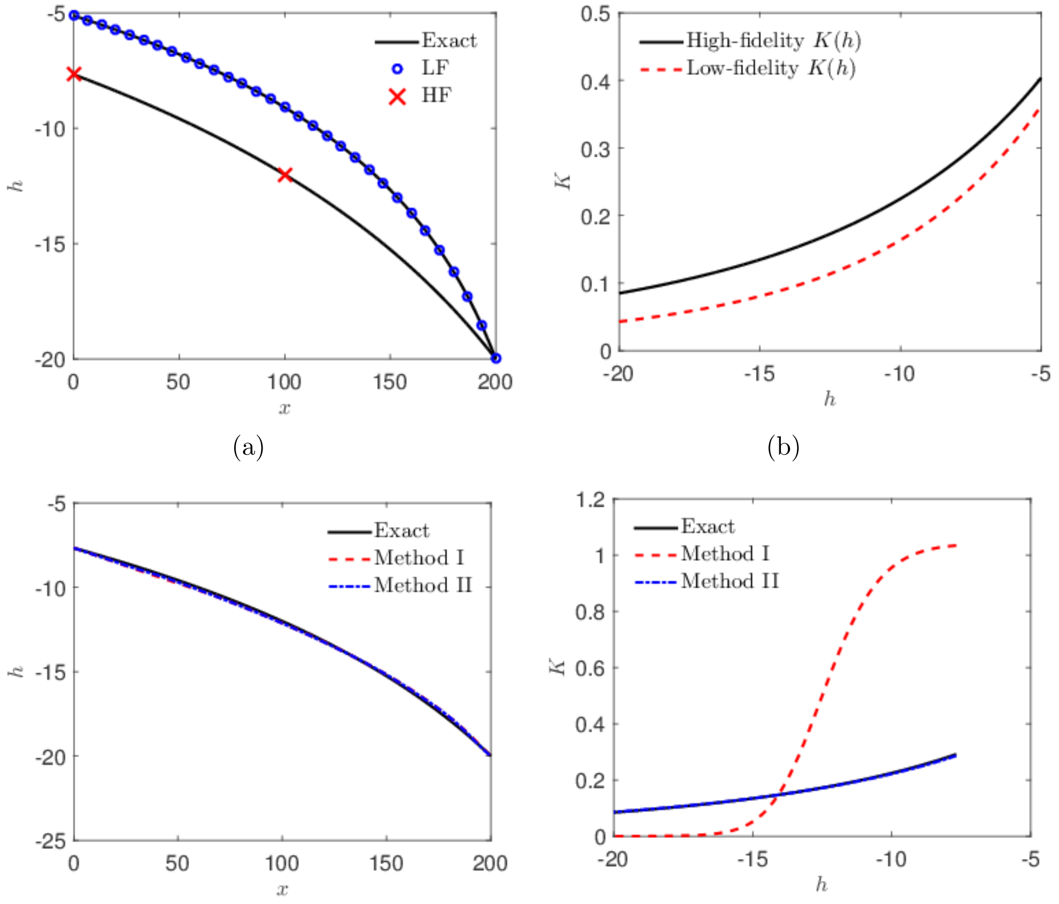
The loam is selected as a representative case here, for which the empirical ranges of α_0 and m are: $\alpha_0(cm^{-1}) \in [0.015, 0.057]$ and $m \in [0.31, 0.40]$ [30]. In addition, $K_s = 1.04cm/hr$. To obtain the training data for neural networks, two types of numerical simulations are conducted to generate the low- and high-fidelity data using the *bvp4c* in *Matlab* (uniform lattice with $\delta_x = 1/15cm$). For high-fidelity data, the exact values for α_0 and m are assumed to be 0.036 cm^{-1} and 0.36. The high-fidelity simulations are then conducted using the exact values of α_0 and m. Different initial guesses for α_0 and m are employed in the low-fidelity simulations. Specifically, ten uniformly distributed pairs i.e., (α_0, m) in the range $(0.015, 0.31) - (0.057, 0.40)$ are adopted in the low-fidelity simulations. For all cases, 31 uniformly distributed sampling data at the low-fidelity level are served as the training data, and only two sampling points including the boundary conditions mentioned above are employed as the training data for high-fidelity. In addition, a smaller learning rate i.e., 10^{-4} is employed for all test cases in this section.

We first consider the flow with constant flux inlet. The flux at the inlet and the pressure at the outlet are set as $q_0 = 0.01cm/y$ and $h_1 = -20cm$, respectively. Equation (22) is added into the last neural network in MPINNs. We employ the numerical results for $\alpha_0 = 0.055$ and $m = 0.4$ as the low-fidelity data. As shown in Fig. 7(d), the prediction for hydraulic conductivity is different from the exact solution. According to Darcy's law, we can rewrite Eq. (22) as

$$q(x) = -K\partial_x h,\ \partial_x q(x) = 0. \tag{25}$$

Considering that $q = q_0$ at the inlet is a constant, we can then obtain the following equation

$$q(x) = -K\partial_x h = q_0, \tag{26}$$



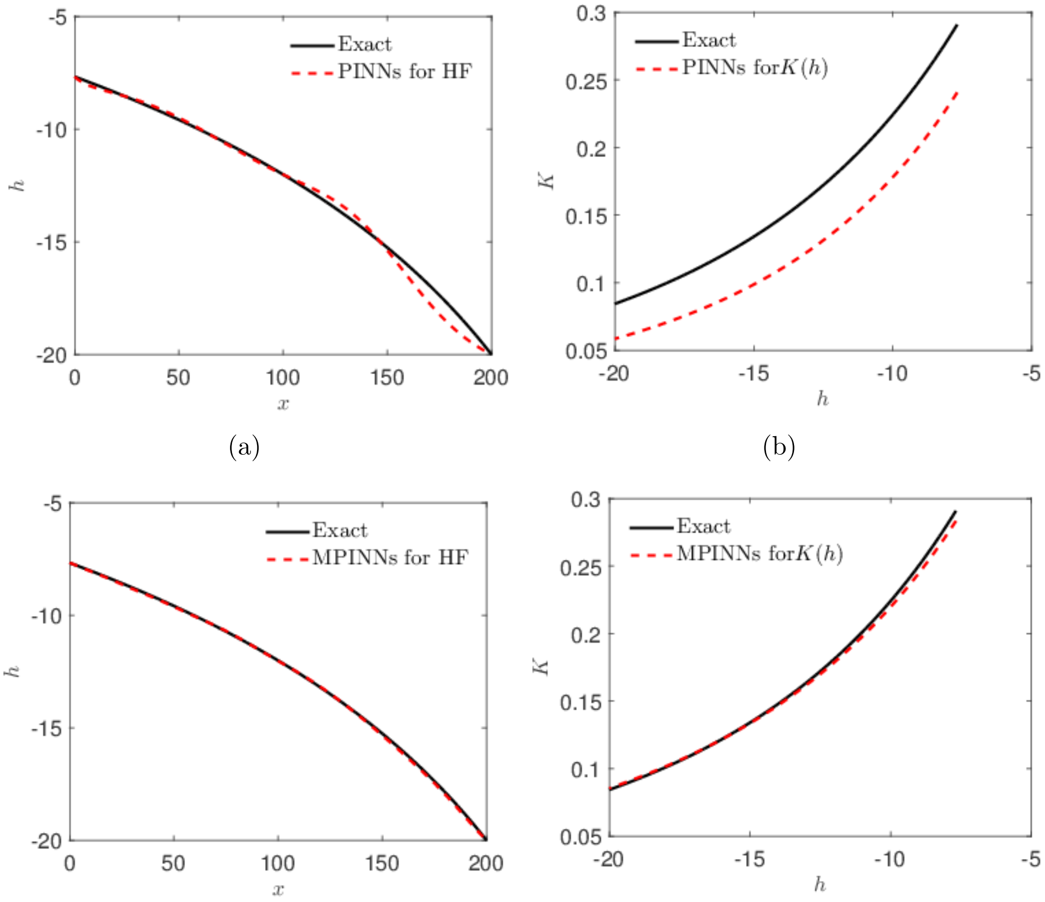
(a)

(b)

(c)

(d)

Figure 7: Predictions for unsaturated flow in porous media using the differential (Eq. (22)) and integral formulations (Eq. (26)) with constant flux at the inlet and constant pressure head at the outlet. (a) Training data for pressure head. (b) Low- and high-fidelity hydraulic conductivity. (c) Predicted pressure head using MPINNs training with multi-fidelity data. Method I: Differential formulation, Method II: Integral formulation. (d) Predicted hydraulic conductivity using MPINNs training with multi-fidelity data. Method I: Differential formulation, Method II: Integral formulation.



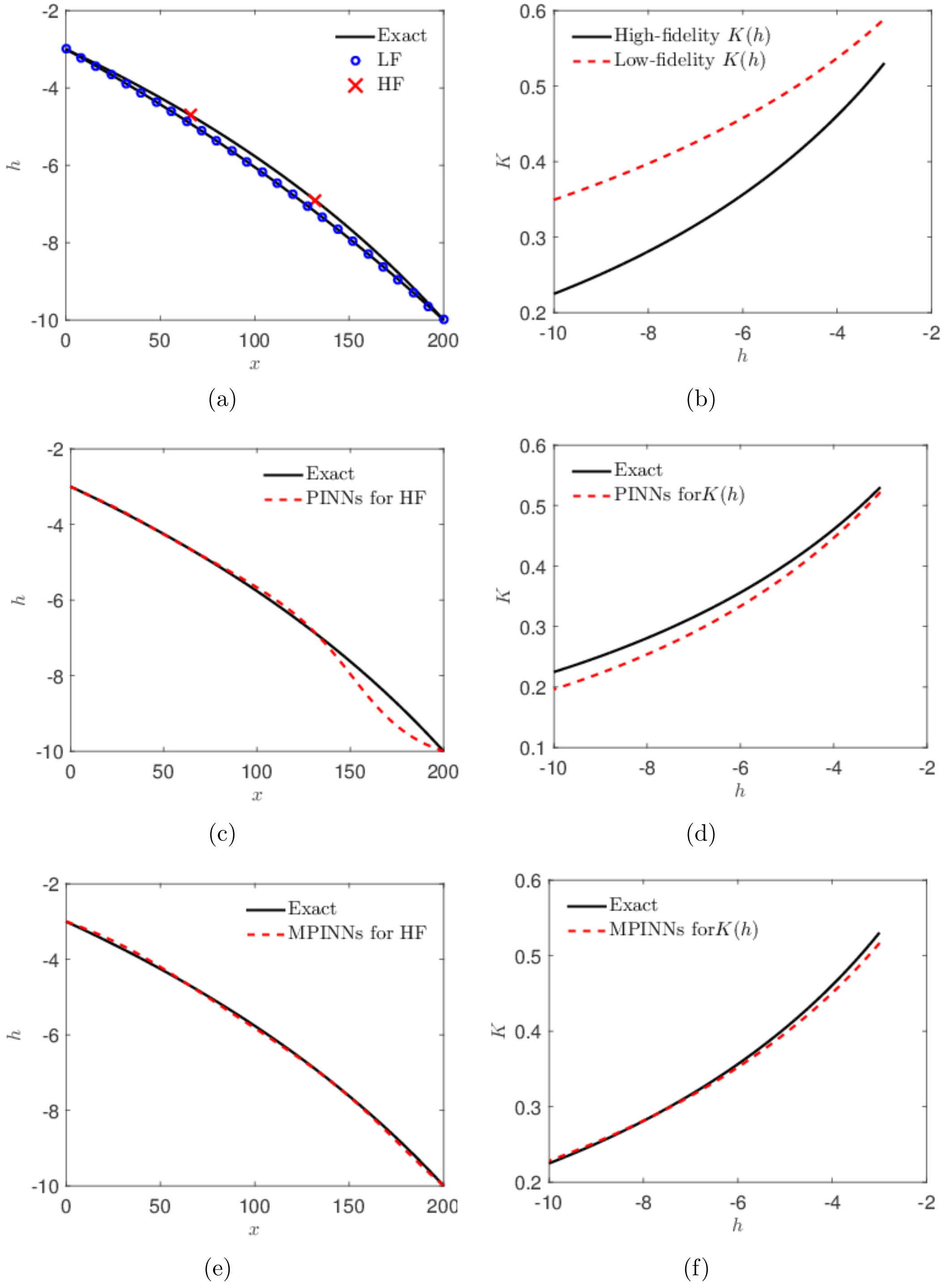
(a)

(b)

(c)

(d)

Figure 8: Predictions for unsaturated flow in porous media using the integral formulation (Eq. (26)) with constant flux at the inlet and constant pressure head at the outlet. (a) Predicted pressure head using PINNs training with high-fidelity data only. (b) Predicted hydraulic conductivity using PINNs training with high-fidelity data only. (c) Predicted pressure head using MPINNs with multi-fidelity data. (d) Predicted hydraulic conductivity using using MPINNs with multi-fidelity data.



(a)

(b)

(c)

(d)

(e)

(f)

Figure 9: Predictions for unsaturated flow in porous media using the integral formulation (Eq. (26)) with constant pressure head at the inlet and outlet. (a) Training data for pressure head. Low-fidelity data is computed with $\alpha_0 = 0.015$ and $m = 0.31$. (b) Low- and high-fidelity hydraulic conductivity. Low-fidelity hydraulic conductivity is computed with $\alpha_0 = 0.015$ and $m = 0.31$. (c) Predicted pressure head using PINNs training with high-fidelity data only. (d) Predicted hydraulic conductivity using PINNs training with high-fidelity data only. (e) Predicted pressure head using MPINNs with multi-fidelity data. (f) Predicted hydraulic conductivity using using MPINNs with multi-fidelity data.

which actually is the mass conservation at each cross section. We then employ Eq. (26) instead of Eq. (22) in the MPINNs, and the results improve greatly (Fig. 7(d)).

We proceed to study this case in some more detail. We perform the single-fidelity modeling (SF) based on the high-fidelity data. We use two hidden layers with 20 neurons per layer in $\mathcal{NN}_{H_2}$, in which the hyperbolic tangent function is employed as the activation function. The learned pressure head and the hydraulic conductivity are shown in Figs. 8(a)-8(b). We observe that both the learned h and $K(h)$ disagree with the exact results. We then switch to multi-fidelity modeling. Two hidden layers and 10 neurons per layer are used in $\mathcal{NN}_L$, and two hidden layers with 10 neurons per layer are utilized for $\mathcal{NN}_{H_2}$. The predicted pressure head as well as the hydraulic conductivity (average value from ten runs with different initial guesses) agree quite well with the exact values (Figs. 8(c)-8(d)). For Case II, we set the pressure head at the inlet and outlet as $h_0 = -3cm$ and $h_1 = -10cm$. We also assume that the flux at the inlet is known, thus Eq. (26) can also be employed instead of Eq. (22) in the MPINNs. The training data are illustrated in Fig. 9(a). The size of the NNs here is kept the same as that used in Case I. We observe that results for the present case (Figs. 9(c)-9(f)) are quite similar with those in Case I.

Finally, the mean values of α_0 as well as the m for different initial guesses are shown in Table 2, which indicates that the MPINNs can significantly improve the prediction accuracy as compared to the estimations based on the high-fidelity only (SF in Table 2).

Table 2: PINN and MPINN predictions for hydraulic conductivity.

	$\alpha_0(cm^{-1})$	$\sigma(\alpha_0)$	m	$\sigma(m)$
SF (Case I)	0.0438	-	0.359	-
MF (Case I)	0.0344	0.0027	0.347	0.0178
SF (Case II)	0.0440	-	0.377	-
MF (Case II)	0.0337	7.91×10^{-4}	0.349	0.0037
Exact	0.036	-	0.36	-

3.2.2. Estimation of reaction models for reactive transport

We further consider a single irreversible chemical reaction in a 1D soil column with a length of $5m$, which is expressed as

$$a_r A \rightarrow B, \tag{27}$$

where A, and B are different solute. The above reactive transport can be described by the following advection-dispersion-reaction equation as

$$\partial_t(\psi C_i) + q\partial_x C_i = \psi D\partial_x^2 C_i - \psi v_i k_{f,r} C_A^{a_r}, (i = A, B), \tag{28}$$

where $C_i(mol/L)$ is the concentration of any solute, q is the Darcy velocity, ψ is the porosity, D is the dispersion coefficient, $k_{f,r}$ denotes the chemical reaction rate, a_r is the order of the chemical reaction, both of which are difficult to measure directly, and v_i is the stoichiometric coefficient with $v_A = a_r$, and $v_B = -1$. Here, we assume that the following parameters are known: $\psi = 0.4$, $q = 0.5m/y$, and $D = 10^{-8}m/s^2$. The initial and boundary conditions imposed on the solute are expressed as

$$C_A(x, 0) = C_B(x, 0) = 0, \tag{29}$$

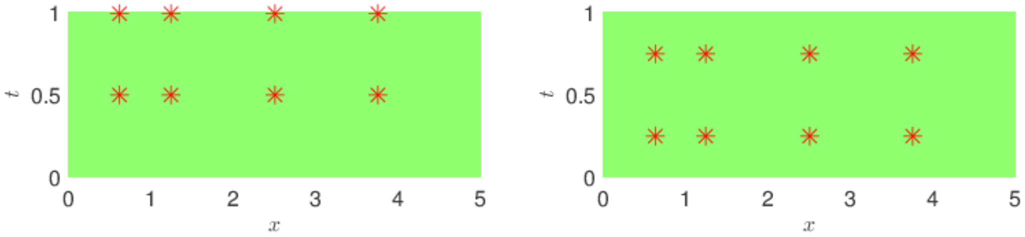
$$C_A(0, t) = 1,\ C_B(0, t) = 0, \tag{30}$$

$$\partial_x C_i(x, t)|_{x=l_x} = 0. \tag{31}$$

The objective here is to learn the effective chemical reaction rate as well as the reaction order based on partial observations of the concentration field $C_A(x, t)$.

We perform lattice Boltzmann simulations [31, 32] to obtain the training data since we have no experimental data. Consider that v_A is a constant, we define an effective reaction rate as $k_f = v_A k_{f,r}$ for simplicity. The exact effective reaction rate and reaction order are assumed to be $k_f = 1.577/y$ and $a_r = 2$, respectively. Numerical simulations with the exact k_f and a_r are then conducted to obtain the high-fidelity data. In simulations, a uniform lattice is employed, i.e., $l_x = 400\delta_x$, where $\delta_x = 0.0125m$ is the space step, and $\delta_t = 6.67 \times 10^{-4}y$ is the time step size. We assume that the sensors for concentration are located at $x = \{0.625, 1.25, 2.5, 3.75\}m$. In addition, we assume that the data are collected from the sensors once half a year. In particular, we employ two different datasets (Fig. 10), i.e., (1) $t = 0.5$ and 1 years (Case I), and (2) $t = 0.25$ and 0.75 years. Schematics of the training data points for the two cases we consider are shown in Fig. 10(a) and Fig. 10(b).

Next, we describe how we obtain the low-fidelity data. In realistic applications, the pure chemical reaction rate (without porous media) between different solute e.g., A and B are known, which can be served as the initial guess for k_f. Here we assume that the initial guess for the chemical reaction rate and reaction order vary from $0.75k_f/a_r$ to $1.25k_f/a_r$. To study the effect of the initial guess $((k_{f,0}, a_{r0}))$ on the predictions, we conduct the low-fidelity numerical simulations based on ten uniformly distributed pairs in $[0.75k_f, 0.75a_r]-[1.25k_f, 1.25a_r]$ using the same grid size and time step as the high-fidelity simulations. Here $k_{f,0}$ and a_{r0} represent the initial guesses for k_f and a_r. The learning rate employed in this section is also 10^{-4}.



(a)

(b)

Figure 10: Schematic of the space-time domain and the locations of the high-fidelity data for modeling reactive transport. (a) Case I: Data are collected at $t = 0.5$ and 1 years. (b) Case II: Data are collected at $t = 0.25$ and 0.75 years.

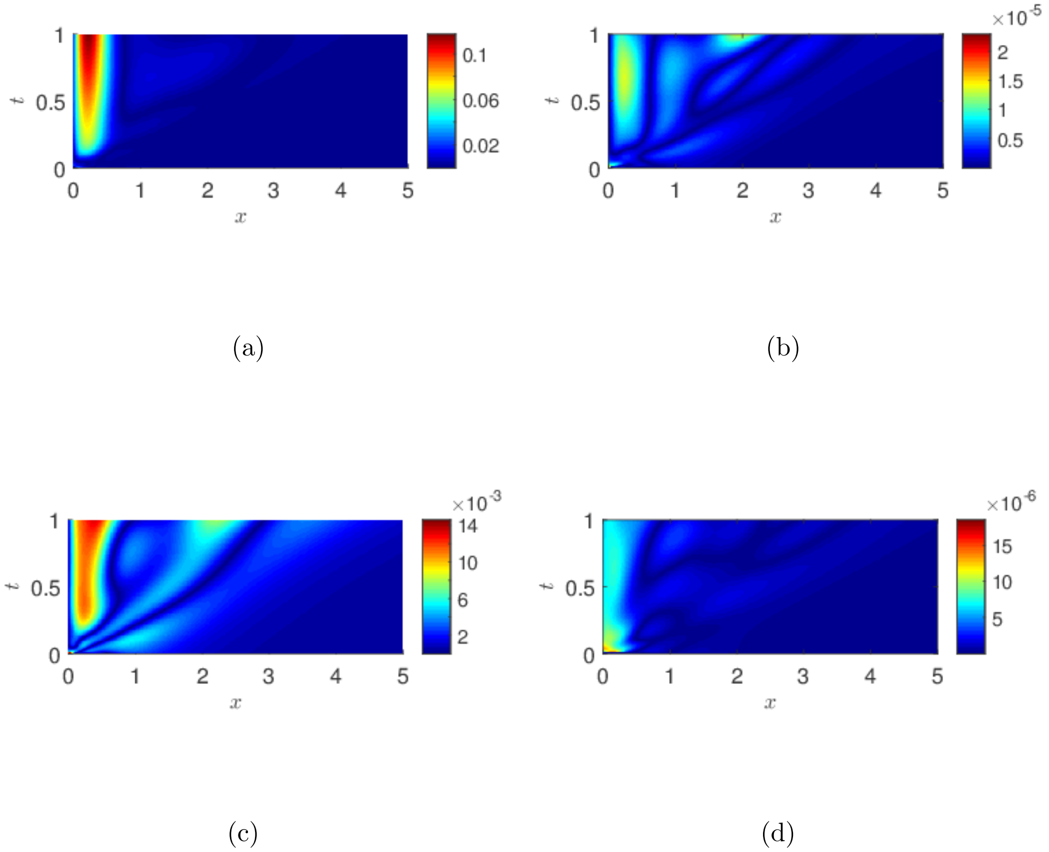
The results of predictions using PINNs (with the hyperbolic tangent activation function) trained on high-fidelity data are shown in Fig. 11(a) and Fig. 11(c) for the two cases we consider, and corresponding results using MPINNs are shown in Fig. 11(b) and Fig. 11(d). The estimated mean and standard deviation for k_f and a_r are displayed in Table 3, which are much better than the results from single-fidelity modelings. We also note that the standard deviations are rather small, which demonstrates the robustness of the MPINNs.

Table 3: PINN and MPINN predictions for reactive transport.

	$k_f(/y)$	$\sigma(k_f)$	a_r	$\sigma(a_r)$
SF (Case I)	0.441	-	0.558	-
MF (Case I)	1.414	7.45×10^{-3}	1.790	9.44×10^{-3}
SF (Case II)	1.224	-	1.516	-
MF (Case II)	1.557	2.14×10^{-2}	1.960	2.57×10^{-2}
Exact	1.577	-	2	-

4. Conclusion

In this work we presented a new composite deep neural network that learns from multi-fidelity data, i.e. a small set of high-fidelity data and a larger set of inexpensive



(a)

(b)

(c)

(d)

Figure 11: Predicted concentration field. (a) Case I: Relative errors (absolute value) using a PINN trained on high-fidelity data only. $\mathcal{NN}_{H_2} \rightarrow 4 \times 20$. (b) Case I: Mean relative errors (absolute value) using a MPINN trained on multi-fidelity data. Initial guesses: ten uniformly distributed pairs in $[0.75k_f, 0.75a] - [1.25k_f, 1.25a]$. The concentration fields plotted are the mean values for ten runs with different initial guesses. $\mathcal{NN}_L \rightarrow 2 \times 10$, $\mathcal{NN}_{H_2} \rightarrow 2 \times 10$. (c) Case II: Relative errors (absolute value) using a PINN trained on high-fidelity data only. $\mathcal{NN}_{H_2} \rightarrow 4 \times 20$. (d) Case II: Mean relative errors (absolute value) using a MPINN trained on multi-fidelity data. $\mathcal{NN}_L \rightarrow 2 \times 10$, $\mathcal{NN}_{H_2} \rightarrow 2 \times 10$.

low-fidelity data. This scenario is prevalent in many cases for modeling physical and biological systems and we expect that the new DNN will provide solutions to many current bottlenecks where availability of large data sets of high-fidelity is simply not possible but either low-fidelity data from inexpensive sensors or other modalities or even simulated data can be readily obtained. Moreover, we extended the concept of physics-informed neural networks (PINNs) that use a single-fidelity data to train to the multi-fidelity case and MPINNs. Specifically, MPINNs are composed of four fully connected neural networks: the fist neural network approximates the low-fidelity data, while the second and third NNs are for constructing the correlations between the low- and high-fidelity data, and the last NN encodes the PDEs that describe the corresponding physical problems. The two sub-networks included in the high-fidelity network are employed to approximate the linear and nonlinear parts of the correlations, respectively. Training the relaxation parameter employed to combine the two sub-networks enables the MPINNs to learn the correlation based on the training data without any prior assumption on the relation between the low- and high-fidelity data.

MPINNs have the following attractive features: (1) Owing to the expressible capability of function approximation of the NNs, multi-fidelity NNs are able to approximate both continuous and discontinuous functions in high dimensions; (2) Due to the fact that NNs can handle almost any kind of nonlinearities, MPINNs are effective for identification of unknown parameters or functions in inverse problems described by nonlinear PDEs.

We first tested the new multi-fidelity DNN in approximating continuous and discontinuous functions with linear and nonlinear correlations. Our results demonstrated that the present model can adaptively learn the correlations between the low- and high-fidelity data based on the training data of variable fidelity. In addition, this model can easily be extended based on the embedding theory to learn more complicated nonlinear and non-functional correlations. We then tested MPINNs on inverse PDE problems, namely, in estimating the hydraulic conductivity for unsaturated flows as well as the reaction models in reactive transport in porous media. We found that the proposed new MPINN can identify the unknown parameters or even functions with high accuracy using very few high-fidelity data, which is promising in reducing the high experimental cost for collecting high-fidelity data. Finally, we point out that MPINNs can also be employed for high-dimensional problems as well as problems with multiple fidelities, i.e. more than two fidelities.

Acknowledgement

This work was supported by the PHILMS grant DE-SC0019453, the DOE-BER grant DE-SC0019434, the AFOSR grant FA9550-17-1-0013, and the DARPA-AIRA grant HR00111990025. In addition, we would like to thank Dr. Guofei Pang, Dr. Zhen Li, Dr. Zhiping Mao, and Ms Xiaoli Chen for their helpful discussions.

Appendix A. Data-driven manifold embeddings

To learn more complicated non-linear correlation between the low- and high-fidelity data, we can further link the multi-fidelity DNNs with the embedding theory [25]. According to the weak Whitney embedding theorem [33], any continuous function from an n-dimensional manifold to an m-dimensional manifold may be approximated by a smooth embedding with $m > 2n$. Using this theorem, Taken's theorem [34] further points out that the m embedding dimensions can be composed of m different observations of the system state variables or m time delays for a single scalar observable.

Now we will introduce the applications of the two theorems in multi-fidelity modelings. We assume that both y_L and y_H are smooth functions. Suppose that $y_L, y_L(x-\tau), ..., y_L(x-(m-1)\tau)$ (τ is the time delay) and a small number of (x, y_H) are available, we can then express y_H in the following form

$$y_H(x) = \mathcal{F}(x, y_L(x), y_L(x - i\tau)), i = 1, ..., m - 1. \tag{A.1}$$

By using this formulation, we can construct more complicate correlation than Eq. (2). To link the multi-fidelity DNN with the embedding theory, we can extend the inputs for $\mathcal{NN}_{H,i}$ to higher dimensions, i.e., $[x, y_L(x)] \to [x, y_L(x), y_L(x-\tau), y_L(x-2\tau), ...,$ $y_L(x-(m-1)\tau)]$, which enables the multi-fidelity DNN to discover more complicated underlying correlations between the low- and high-fidelity data.

Note that the selection of optimal value for the time delay τ is important in embedding theory [35–37], on which numerous studies have been carried out [35]. However, most of the existing methods for determining the optimal value of τ appear to be problem-dependent [35]. Recently, Dhir *et al.* proposed a Bayesian delay embedding method, where τ is robustly learned from the training data by employing the variational autoencoder [37]. In the present study, the value of τ is also learned by optimizing the NNs rather than setting it as constant as in the original work presented in Ref. [25].

References

[1] N. M. Alexandrov, R. M. Lewis, C. R. Gumbert, L. L. Green, P. A. Newman, Approximation and model management in aerodynamic optimization with variable-fidelity models, J. Aircraft 38 (6) (2001) 1093–1101.

[2] D. Böhnke, B. Nagel, V. Gollnick, An approach to multi-fidelity in conceptual aircraft design in distributed design environments, in: 2011 Aerospace Conference, IEEE, 2011, pp. 1–10.

[3] L. Zheng, T. L. Hedrick, R. Mittal, A multi-fidelity modelling approach for evaluation and optimization of wing stroke aerodynamics in flapping flight, J. Fluid Mech. 721 (2013) 118–154.

[4] N. V. Nguyen, S. M. Choi, W. S. Kim, J. W. Lee, S. Kim, D. Neufeld, Y. H. Byun, Multidisciplinary unmanned combat air vehicle system design using multi-fidelity model, Aerosp. Sci. and Technol. 26 (1) (2013) 200–210.

[5] M. C. Kennedy, A. O'Hagan, Predicting the output from a complex computer code when fast approximations are available, Biometrika 87 (1) (2000) 1–13.

[6] A. I. Forrester, A. Sóbester, A. J. Keane, Multi-fidelity optimization via surrogate modelling, Proc. Math. Phys. Eng. Sci. 463 (2088) (2007) 3251–3269.

[7] M. Raissi, G. E. Karniadakis, Deep multi-fidelity Gaussian processes, arXiv preprint arXiv:1604.07484, 2016.

[8] M. Raissi, P. Perdikaris, G. E. Karniadakis, Inferring solutions of differential equations using noisy multi-fidelity data, J. Comput. Phys. 335 (2017) 736–746.

[9] P. Perdikaris, M. Raissi, A. Damianou, N. Lawrence, G. E. Karniadakis, Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling, Proc. Math. Phys. Eng. Sci. 473 (2198) (2017) 20160751.

[10] L. Bonfiglio, P. Perdikaris, G. Vernengo, J. S. de Medeiros, G. E. Karniadakis, Improving swath seakeeping performance using multi-fidelity Gaussian process and Bayesian optimization, J. Ship Res. 62 (4) (2018) 223–240.

[11] K. J. Chang, R. T. Haftka, G. L. Giles, I. J. Kao, Sensitivity-based scaling for approximating structural response, J. Aircraft 30 (2) (1993) 283–288.

[12] R. Vitali, R. T. Haftka, B. V. Sankar, Multi-fidelity design of stiffened composite panel with a crack, Struct. and Multidiscip. O. 23 (5) (2002) 347–356.

[13] M. Eldred, Recent advances in non-intrusive polynomial chaos and stochastic collocation methods for uncertainty analysis and design, in: 50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference 17th AIAA/ASME/AHS Adaptive Structures Conference 11th AIAA No, 2009, p. 2274.

[14] A. S. Padron, J. J. Alonso, M. S. Eldred, Multi-fidelity methods in aerodynamic robust optimization, in: 18th AIAA Non-Deterministic Approaches Conference, 2016, p. 0680.

[15] J. Laurenceau, P. Sagaut, Building efficient response surfaces of aerodynamic functions with Kriging and Cokriging, AIAA J. 46 (2) (2008) 498–507.

[16] E. Minisci, M. Vasile, Robust design of a reentry unmanned space vehicle by multi-fidelity evolution control, AIAA J. 51 (6) (2013) 1284–1295.

[17] P. Lancaster, K. Salkauskas, Surfaces generated by moving least squares methods, Math. Comput. 37 (155) (1981) 141–158.

[18] D. Levin, The approximation power of moving least-squares, Math. Comp. 67 (224) (1998) 1517–1531.

[19] M. G. Fernández-Godino, C. Park, N. H. Kim, R. T. Haftka, Review of multifidelity models, arXiv preprint arXiv:1609.07196, 2016.

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

[21] Q. Zheng, J. Zhang, W. Xu, L. Wu, L. Zeng, Adaptive multi-fidelity data assimilation for nonlinear subsurface flow problems, Water Resour. Res. 55 (2018) 203–217.

[22] 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, J. Comput. Phys. 378 (2019) 686–707.

[23] M. Raissi, A. Yazdani, G. E. Karniadakis, Hidden fluid mechanics: A Navier-Stokes informed deep learning framework for assimilating flow visualization data, arXiv preprint arXiv:1808.04327, 2018.

[24] A. M. Tartakovsky, C. O. Marrero, D. Tartakovsky, D. Barajas-Solano, Learning parameters and constitutive relationships with physics informed deep neural networks, arXiv preprint arXiv:1808.03398, 2018.

[25] S. Lee, F. Dietrich, G. E. Karniadakis, I. G. Kevrekidis, Linking Gaussian process regression with data-driven manifold embeddings for nonlinear data fusion, arXiv preprint arXiv:1812.06467, 2018.

[26] S. Shan, G. G. Wang, Metamodeling for high dimensional simulation-based design problems, J. Mech. Design 132 (5) (2010) 051009.

[27] S. L. Markstrom, R. G. Niswonger, R. S. Regan, D. E. Prudic, P. M. Barlow, Gsflow-coupled ground-water and surface-water flow model based on the integration of the precipitation-runoff modeling system (prms) and the modular ground-water flow model (modflow-2005), US Geological Survey techniques and methods 6 (2008) 240.

[28] M. Hayashi, D. O. Rosenberry, Effects of ground water exchange on the hydrology and ecology of surface water, Groundwater 40 (3) (2002) 309–316.

[29] M. T. Van Genuchten, A closed-form equation for predicting the hydraulic conductivity of unsaturated soils, Soil Sci. Soc. Am. J. 44 (5) (1980) 892–898.

[30] R. F. Carsel, R. S. Parrish, Developing joint probability distributions of soil water retention characteristics, Water Resour. Res. 24 (5) (1988) 755–769.

[31] X. Meng, Z. Guo, Localized lattice Boltzmann equation model for simulating miscible viscous displacement in porous media, J. Heat Mass Trans. 100 (2016) 767–778.

[32] B. Shi, Z. Guo, Lattice Boltzmann model for nonlinear convection-diffusion equations, Phys. Rev. E 79 (1) (2009) 016701.

[33] H. Whitney, Differentiable manifolds, Ann. Math. (1936) 645–680.

[34] F. Takens, Detecting strange attractors in turbulence, in: Dynamical systems and turbulence, Warwick 1980, Springer, 1981, pp. 366–381.

[35] R. Hegger, H. Kantz, T. Schreiber, Practical implementation of nonlinear time series methods: The tisean package, Chaos 9 (2) (1999) 413–435.

[36] H. D. Abarbanel, R. Brown, J. J. Sidorowich, L. S. Tsimring, The analysis of observed chaotic data in physical systems, Rev. Mod. Phys. 65 (4) (1993) 1331.

[37] N. Dhir, A. R. Kosiorek, I. Posner, Bayesian delay embeddings for dynamical systems, Conference on Neural Information Processing Systems, 2017.