Mini-data-driven Deep Arbitrary Polynomial Chaos Expansion for Uncertainty Quantification

Xiaohu Zheng\textsuperscript{a,b}, Jun Zhang\textsuperscript{b}, Ning Wang\textsuperscript{b}, Guijian Tang\textsuperscript{b}, Wen Yao\textsuperscript{b,*}

\textsuperscript{a}College of Aerospace Science and Engineering, National University of Defense Technology, No. 109, Deya Road, Changsha 410073, China
\textsuperscript{b}National Innovation Institute of Defense Technology, Chinese Academy of Military Science, No. 53, Fengtai East Street, Beijing 100071, China

\textbf{Abstract}

The surrogate model-based uncertainty quantification method has drawn a lot of attention in recent years. Both the polynomial chaos expansion (PCE) and the deep learning (DL) are powerful methods for building a surrogate model. However, the PCE needs to increase the expansion order to improve the accuracy of the surrogate model, which causes more labeled data to solve the expansion coefficients, and the DL also needs a lot of labeled data to train the neural network model. This paper proposes a deep arbitrary polynomial chaos expansion (Deep aPCE) method to improve the balance between surrogate model accuracy and training data cost. On the one hand, the multilayer perceptron (MLP) model is used to solve the adaptive expansion coefficients of arbitrary polynomial chaos expansion, which can improve the Deep aPCE model accuracy with lower expansion order. On the other hand, the adaptive arbitrary polynomial chaos expansion’s properties are used to construct the MLP training cost function based on only a small amount of labeled data and a large scale of non-labeled data, which can significantly reduce the training data cost. Four numerical examples and an actual engineering problem are used to verify the effectiveness of the Deep aPCE method.

\textit{Keywords:} Mini-data, Arbitrary polynomial chaos expansion, Deep learning, Uncertainty quantification

1. Introduction

In many engineering fields like mechanical engineering [1, 2], aerospace engineering [3, 4], and civil engineering [5, 6], some uncertainty factors such as environmental changes, manufacture defects of mechanical structures will lead to uncertainties in the physical parameters, the external loads, and the boundary conditions [6], etc. The above uncertainties will significantly influence the performance of the system. Therefore, uncertainty quantification is essential for analyzing the effects of uncertainties on the system performance and minimize the risk of system failure [7, 8].

In recent years, the uncertainty quantification methods mainly include the sampling-based simulation method (e.g., Monte Carlo simulation [9]), the surrogate model-based method (e.g., polynomial chaos expansion [10]), the Taylor series expansion-based method (e.g., the first-order and second-order reliability method [11]), etc. For the surrogate model-based method, it constructs a simple and explicit surrogate model to replace the...
original complicated and computationally expensive model like finite element analysis (FEA). Once the explicit
surrogate model is constructed, the researcher can straightforwardly perform the Monte Carlo simulation
(MCS) on the surrogate model with a lower computational cost. Then, any probabilistic quantity of interest can
be obtained according to the results of MCS. Thus, some surrogate model-based methods like the polynomial
chaos expansion (PCE) [12, 13], deep learning method [14, 15], response surface method [16, 17], Kriging
method [18, 19], support vector machine [20, 21] have drawn a lot of attention for quantifying uncertainty
in recent years. By expanding the stochastic system’s output response onto a basis composed of orthogonal
polynomials, PCE becomes a versatile and powerful method for building a surrogate model.

For PCE, it expands the stochastic system model into a polynomial model based on the polynomial
orthogonal basis of random input variables. By employing the Hermite polynomials, Wiener [22] introduced
the original Wiener–Hermite expansion firstly in 1938. For the stochastic system model with normal random input
variables, the original Wiener–Hermite expansion is a very effective method. However, if the stochastic system
model’s random input variables are non-normal variables, the original Wiener–Hermite expansion shows a
prolonged convergence rate. To solve this problem, Xiu et al. [10] proposed generalized polynomial chaos
(gPC) by employing orthogonal polynomial functionals from the Askey scheme. The gPC makes PCE suitable
for the stochastic system model whose random input variables obey parametric statistical distributions like
Gamma, Beta, Uniform, etc. Unfortunately, information about the distribution of data is very limited in
practical engineering applications, and this will make the parametric statistical distributions of random input
variables unknown. Thereby, the gPC is not suitable for this situation. To make PCE suitable for a larger
spectrum of distributions, Oladyshkin et al. [12] proposed the arbitrary polynomial chaos expansion (aPC)
method based on a finite number of moments of random input variables. The main feature of aPC method is
that it allows the researchers to select freely technical constraints the shapes of their statistical assumptions.
After constructing the PCE model by the above three methods, the expansion coefficients can be solved by
Galerkin projection method [10, 23, 24], collocation method [25, 26, 27], etc. Generally, the PCE model’s
accuracy can be improved by increasing the expansion order. However, this increases the number of samples for solving the expansion coefficients, which will result in higher computational
cost in constructing the PCE model.

Besides, the PCE method suffers the problem known as the ‘curse of dimensionality’ for high dimensional
stochastic systems, i.e., the number of expansion coefficients increases dramatically with the dimension of
random input variables. To overcome this problem, Xu et al. [5] decomposed the original performance function to be a summation of several component functions by contribution-degree analysis, and then the gPC
is employed to reconstruct these component functions. Pan et al. [28] adopted the sliced inverse regression
technique to achieve a dimension reduction, then the sparse PCE was used to construct the surrogate model.
Blatman et al. [29] proposed an adaptive regression-based algorithm that can automatically detect the significant coefficients of the PCE. As a consequence, a relatively small number of expansion coefficients is eventually retained. Besides, an adaptive algorithm based on least angle regression was also proposed for automatically
detecting the significant coefficients of PCE by Blatman et al. [30]. For the dimension-reduction steps or the significant coefficients detecting in the above methods, however, it not only complicates
the process of building a PCE model but also will affect the accuracy of the PCE model to a certain extent.

To simulate a high dimensional stochastic system, the deep learning (DL) [31] is applied to construct a surrogate model, and the deep feedforward network is a classical DL model [32]. According to the researches by Hornik et al. [33] and Cybenko [34], the surrogate model with any desired degree of accuracy can be obtained by the deep feedforward network. Thus, the deep feedforward network can handle the ‘curse of dimensionality’ by multiple nonlinear activation functions [35]. In recent years, the deep feedforward network has been studied in many fields for constructing surrogate model [35, 36, 37, 38]. By dividing the hot wall into multiple domains, Shahane et al. [35] used sets of forward deterministic simulations to train deep neural networks. Based on a deep neural network, Lye et al. [37] proposed an iterative surrogate model optimization method whose key feature is the iterative selection of training data. By blending different fidelity information, Zhang et al. [38] constructed a high-accuracy multi-fidelity surrogate model for designing aircraft aerodynamic shape based on a deep neural network. Refer to the above methods, the deep feedforward network can build a high-precision surrogate model. However, DL’s main defect is that a large number of labeled training data need to be used to train the deep neural network, which will lead to the unacceptable computational cost.

In summary, PCE and DL have the following three shortcomings: Firstly, the PCE model’s accuracy can be improved by increasing the expansion order, but the computational cost will increase accordingly; Secondly, for higher dimensional stochastic systems, the existed PCE solution methods not only complicate the construction process of the PCE model but also affect the PCE model’s accuracy to a certain extent; Thirdly, the DL can construct a high-precision surrogate model but needs a lot of labeled training data to train the deep neural network. Therefore, to improve the PCE model’s accuracy and reduce the number of samples for solving the expansion coefficients, this paper proposes a deep arbitrary polynomial chaos expansion (Deep aPCE) method by making full use of the advantages of aPC and DL. For the Deep aPCE method, the aPC is used to construct a $p$-order PCE model. Different from the traditional PCE methods [10, 12, 22], the expansion coefficients of the Deep aPCE model are adaptive expansion coefficients which are fine-tuned dynamically by the MLP model for different random input variables. To reduce the calling times of computationally expensive models, the Deep aPCE method adopts two kinds of training data, i.e., a small amount of labeled training data and some non-labeled training data. For the non-labeled training data, the researchers only need to generate random input data of the stochastic system model. Thus, the computational cost of preparing non-labeled training data is meager. Then, a cost function is proposed based on the properties of the adaptive arbitrary polynomial chaos expansion for training the Deep aPCE model. Compared with the existed PCE method, the Deep aPCE method needs fewer calling times of computationally expensive models to build a higher accuracy surrogate model. What’s more, due to the application of DL, the proposed Deep aPCE method can directly construct accurate surrogate models for the appropriate high dimensional stochastic systems without any dimension-reduction processing. After training the Deep aPCE model, the researchers
can straightforwardly perform MCS on the Deep aPCE model with a low computational cost. Thereby, any probabilistic quantity of interest can be obtained according to the results of MCS.

The rest of this paper is organized as follows. In section 2, the theoretical bases including MLP, aPC and error estimates are introduced. Then, the proposed Deep aPCE method are presented in section 3: In section 3.1 and section 3.2, the framework of Deep aPCE method and the Deep aPCE model modeling method are proposed respectively; In section 3.3, the cost function are proposed to learning the parameters of Deep aPCE model; The Deep aPCE model training algorithm is proposed in section 3.4; In section 3.5, the expansion order are discussed. In section 4, four numerical examples are used to verify the effectiveness of the proposed Deep aPCE method. Finally, the proposed Deep aPCE method is applied to the uncertainty analysis for the first-order frequency of micro-satellite TianTuo-3 frame structure in section 5.

2. Theoretical bases

2.1. Multilayer perceptron

Definition of MLP. MLP is also called deep feedforward network, which is a classical DL model [32]. An MLP consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. Except for the input nodes, each node is a neuron. Besides, each node in the hidden layer uses a nonlinear activation function [39]. As shown in Fig.1, it is a \((L + 2)\) layers MLP.

![Figure 1: An MLP model with \((L + 2)\) layers](image-url)

For the \(j_i\)th \((j_i = 1, 2, \cdots, n_i)\) neuron in the \(i\)th \((i = 1, 2, \cdots, L)\) hidden layers, its value \(h_{j_i}^i\) can be calculated by

\[
h_{j_i}^i = g \left( h_{j_i}^{i-1} w_{j_i,1}^i + h_{j_i}^{i-1} w_{j_i,2}^i + \cdots + h_{j_i}^{i-1} w_{j_i,n_{i-1}}^i + b_{j_i}^i \right) = g \left( h_{j_i}^{i-1} w_{j_i}^i + b_{j_i}^i \right),
\]

where \(g\) is the nonlinear activation function (eg. ReLU\((x)\), sigmoid\((x)\), tanh\((x)\), etc.), \(h_{j_i}^{i-1} = \left[ h_{j_i}^{i-1}, h_{j_i}^{i-1}, \cdots, h_{j_i}^{i-1} \right]
\]

are the values of the \((i-1)\)th hidden layer’s neurons, \(w_{j_i}^i = \left[ w_{j_i,1}^i, w_{j_i,2}^i, \cdots, w_{j_i,n_{i-1}}^i \right]^T\) are the weights, and \(b_{j_i}^i\) is the bias. When \(i = 1\), \(h^0\) is the input of MLP, i.e.

\[
h^0 = [x_1, x_2, \cdots, x_n].
\]
Therefore, all neurons’ values in the $j$th layer are

\[
\begin{align*}
    h^1_i &= g(h^{i-1}_i w^1_i + b^1_i) \\
    h^2_i &= g(h^{i-1}_i w^2_i + b^2_i) \\
    &\vdots \\
    h^j_i &= g(h^{i-1}_i w^j_i + b^j_i) \\
    &\vdots \\
    h^n_i &= g(h^{i-1}_i w^n_i + b^n_i).
\end{align*}
\]  

(3)

Alternatively, Eq.(3) can be written in the more convenient matrix form:

\[
H_{1 \times n_i} = g \left( H_{1 \times n_{i-1}} W_{n_{i-1} \times n_i} + b_{1 \times n_i} \right)
\]

\[
\begin{align*}
    H_{1 \times n_{i-1}} &= \begin{bmatrix} h^1_{i-1} & h^2_{i-1} & \cdots & h^{n_{i-1}}_{i-1} \end{bmatrix} \\
    H_{1 \times n_{i-1}} &= \begin{bmatrix} h^1_i & h^2_i & \cdots & h^{n_i}_i \end{bmatrix} \\
    W_{n_{i-1} \times n_i} &= \begin{bmatrix} w^1_i & w^2_i & \cdots & w^n_i \end{bmatrix} \\
    b_{1 \times n_i} &= \begin{bmatrix} b^1_i & b^2_i & \cdots & b^n_i \end{bmatrix}.
\end{align*}
\]  

(4)

For the output layer, the $k$th ($k = 1, 2, \cdots, m$) neuron’s value is

\[
\hat{y}_k = h^L_1 w^1_k + h^L_2 w^2_k + \cdots + h^L_{n_L} w^k_{n_L} + b_k = h^L w_k + b_k.
\]  

(5)

In Eq.(5), $h^L = [h^L_1, h^L_2, \cdots, h^L_{n_L}]$ are the values of the final hidden layer’s neurons, $b_k$ is the bias, and $w_k = [w^1_k, w^2_k, \cdots, w^k_{n_L}]^T$ is the weights of output layer. Thus, all the outputs of MLP are

\[
\begin{align*}
    \hat{y}_1 &= h^L w_1 + b_1 \\
    \hat{y}_2 &= h^L w_2 + b_2 \\
    &\vdots \\
    \hat{y}_k &= h^L w_k + b_k \\
    &\vdots \\
    \hat{y}_m &= h^L w_m + b_m.
\end{align*}
\]  

(6)

Alternatively, Eq.(6) can be written in the more convenient matrix form:

\[
\begin{align*}
    \hat{y}_{1 \times m} &= H_{1 \times n_L} W_{n_L \times m} + b_{1 \times m} \\
    \hat{y}_{1 \times m} &= \begin{bmatrix} \hat{y}_1 & \hat{y}_2 & \cdots & \hat{y}_m \end{bmatrix} \\
    H_{1 \times n_L} &= \begin{bmatrix} h^L_1 & h^L_2 & \cdots & h^L_{n_L} \end{bmatrix} \\
    W_{n_L \times m} &= \begin{bmatrix} w^1_m & w^2_m & \cdots & w^m_m \end{bmatrix} \\
    b_{1 \times m} &= \begin{bmatrix} b_1 & b_2 & \cdots & b_m \end{bmatrix}.
\end{align*}
\]  

(7)

The goal of MLP is to approximate some function $y = f(x)$. Given the input $x$, the MLP will define a mapping $\hat{y} = \hat{f}(x; \theta)$ can approximate $y = f(x)$ as accurately as possible by learning the parameters $\theta = (W, b)$. 

15
**Parameters learning.** In forward propagation, the lth \((l = 1, 2, \cdots , N)\) input \(x_l\) is propagated to neurons in each layer, and the corresponding final output \(\hat{y}_l\) of MLP can be obtained. Thus, the absolute error \(L (x_l, y_l; \theta)\) between the estimated value \(\hat{y}_l\) and the true value \(y_l\) is

\[
L (x_l, y_l; \theta) = |\hat{y}_l - y_l| = |f (x_l; \theta) - y_l|.
\]  

(8)

Based on \(N\) training data \(\{(x_l, y_l) | l = 1, 2, \cdots , N\}\), the cost function \(J (\theta)\) can be calculated by

\[
J (\theta) = \frac{1}{N} \sum_{l=1}^{N} L (x_l, y_l; \theta).
\]

(9)

Therefore, the gradient \(G (\theta)\) of the cost function \(J (\theta)\) is

\[
G (\theta) = \frac{\partial J (\theta)}{\partial \theta} = \frac{1}{N} \sum_{l=1}^{N} \frac{\partial L (x_l, y_l; \theta)}{\partial \theta}.
\]

(10)

Apparently, the training objective is to minimize the cost function \(J (\theta)\). During the training process, the model parameters \(\theta\) will be updated iteratively as follows:

\[
\theta \leftarrow \theta + \Delta [\eta, G (\theta)],
\]

(11)

where \(\eta\) is the learning rate, \(\Delta (\cdot)\) is the calculation update operator which is determined by the deep neural network optimization algorithm like Adam algorithm [40], AdaGrad algorithm [41], SGD algorithm [42], etc.

2.2. Arbitrary polynomial chaos expansion

**aPC model.** Defined the model \(Y = f (\xi)\) is a stochastic model with the random variable \(\xi\) and the model output \(Y\), where the random variable \(\xi = \{\xi_k | k = 1, 2, \cdots , d\}\). For the stochastic analysis of \(Y\), the model \(f (\xi)\) can be approximated by the p-order aPC model [12, 13], i.e.,

\[
Y \approx g^{(p)} (\xi) = \sum_{i=1}^{M} c_i \Phi_i (\xi),
\]

(12)

where \(c_i\) are the expansion coefficients which can be obtained by Galerkin projection method [10, 23, 24] or collocation method [25, 26, 27], the number of the expansion coefficients \(c_i\) is \(M = (d + p)!/(d!p!)\), and \(\Phi_i (\xi)\) are the multi-dimensional polynomials forming the multi-dimensional orthogonal basis \(\{\Phi_1 (\xi), \Phi_2 (\xi), \cdots , \Phi_M (\xi)\}\).

For the ith multi-dimensional polynomial \(\Phi_i (\xi)\), it is the product of the univariate polynomials of random variables \(\xi_k (k = 1, 2, \cdots , d)\), i.e.,

\[
\Phi_i (\xi) = \prod_{k=1}^{d} \phi_{s_k}(\xi_k),
\]

(13)

\[
\sum_{k=1}^{d} s_k^i \leq p, \quad i = 1, 2, \cdots , M,
\]

where \(s_k^i\) is a multivariate index that contains the individual univariate basis combinatoric information [12], and \(\phi_{s_k}(\xi_k)\) are the univariate polynomials forming the basis \(\{\phi_{(0)}(\xi_k), \phi_{(1)}(\xi_k), \cdots , \phi_{(p)}(\xi_k)\}\) that is orthogonal with respect to the probability measure \(\Gamma [43]\), i.e.,

\[
\int_{\xi_k \in \Omega} \phi_{(j)} (\xi_k) \phi_{(j')}(\xi_k) d \Gamma (\xi_k) = 0, \quad \forall j \neq j', \quad j = 0, 1, \cdots , p,
\]

(14)
where $\Omega$ is the space of events [43]. Different with the Wiener PCE [22, 44, 45] and the gPC [10, 46], the univariate orthogonal basis $\{\phi_k^{(0)}(\xi_k), \phi_k^{(1)}(\xi_k), \cdots, \phi_k^{(p)}(\xi_k)\}$ can be constructed by the raw moment of the random variable $\xi_k$, where the probability measure $\Gamma$ can be arbitrary form.

**Constructing orthogonal basis.** For the $k$th random variable $\xi_k$, the $j$-degree ($j = 0, 1, \cdots, p$) polynomial $\phi_k^{(j)}(\xi_k)$ is defined to be

$$\phi_k^{(j)}(\xi_k) = \sum_{m=0}^{j} a_{m}^{(j)}(\xi_k)^m. \tag{15}$$

where $a_{m}^{(j)} (j = 0, 1, \cdots, p)$ are the coefficients of $\phi_k^{(j)}(\xi_k)$. According to Eqs.(14) and (15), the coefficients $a_{m}^{(j)} (j = 0, 1, \cdots, p)$ can be calculated by the following equation (The detailed derivation process can be found in the reference [12].):

$$
\begin{bmatrix}
\mu_{\xi_k}^{(0)} \\
\mu_{\xi_k}^{(1)} \\
\vdots \\
\mu_{\xi_k}^{(j-1)} \\
0 \\
0 \\
\vdots \\
1
\end{bmatrix}
\begin{bmatrix}
\mu_{\xi_k}^{(0)} \\
\mu_{\xi_k}^{(1)} \\
\vdots \\
\mu_{\xi_k}^{(j-1)} \\
0 \\
0 \\
\vdots \\
1
\end{bmatrix}
= 
\begin{bmatrix}
a_{0}^{(j)} \\
a_{1}^{(j)} \\
\vdots \\
a_{j-1}^{(j)} \\
a_{j}^{(j)}
\end{bmatrix}, \tag{16}
$$

where $\mu_{\xi_k}^{(j)}$ is the $j$th raw moment of random variable $\xi_k$, i.e.

$$\mu_{\xi_k}^{(j)} = \int_{\xi_k \in \Omega} (\xi_k)^j d\Gamma(\xi_k), \tag{17}$$

where $(\xi_k)^j$ denotes $\xi_k$ to the power $j$.

To simplify the explicit form of $a_{m}^{(j)} (j = 0, 1, \cdots, p)$, the random variable $\xi_k$ will be normalized, i.e.,

$$\xi'_k = \frac{\xi_k - \mu_k}{\sigma_k}, \tag{18}$$

where $\mu_k$ and $\sigma_k$ are the mean and the standard deviation of random variable $\xi_k$, respectively. Apparently, normalization will make all moments to be centralize and standardize. To simplify the formula notation, the following content of this paper assumes that $\xi = \{\xi_k | k = 1, 2, \cdots, d\}$ are already a normalized random variable.

Based on Eqs. (16) and (18), the coefficients for polynomials with different degrees can be obtained. For example, the coefficients for polynomials of the 0, 1st, 2nd and 3rd degree are shown in Table 1. According to Eqs.(13) and (15), the multi-dimensional orthogonal basis $\{\Phi_1(\xi), \Phi_2(\xi), \cdots, \Phi_M(\xi)\}$ can be constructed.

In section 3.2.1, a simple method is proposed for constructing the multi-dimensional orthogonal basis.

**Properties of PCE.** The PCE has the following two important properties [12]:

**Property I.** The first coefficient $c_1$ in Eq.(12) is equal to the mean of model output $Y$, i.e.,

$$\mu_Y = c_1. \tag{19}$$

**Property II.** In addition to the first coefficient $c_1$, the sum of squares of coefficients in Eq.(12) is equal to the variance of model output $Y$, i.e.,

$$\sigma_Y^2 = \sum_{i=2}^{M} (c_i)^2. \tag{20}$$
Table 1: Coefficients for polynomials of the 0, 1st, 2nd and 3rd degree

| Degree | Coefficients                                                                 |
|--------|------------------------------------------------------------------------------|
| 0      | $a_0^{(0)} = 1$                                                              |
| 1      | $a_0^{(1)} = 0$, $a_1^{(1)} = 1$                                             |
| 2      | $a_0^{(2)} = -1$, $a_1^{(2)} = -\mu_3^{(3)}$, $a_2^{(2)} = 1$               |
| 3      | $a_0^{(3)} = \left(\mu_3^{(3)}\right)^2 - \left(\mu_3^{(3)}\right)^3 + \mu_3^{(4)} \mu_5^{(4)} - \mu_7^{(5)}$ |
|        | $a_1^{(3)} = -\mu_3^{(3)} \mu_5^{(4)} + \left(\mu_3^{(3)}\right)^2 - \mu_4^{(4)} + \mu_5^{(3)} \mu_5^{(4)}$ |
|        | $a_2^{(3)} = -\mu_3^{(3)} \mu_5^{(4)} + \mu_5^{(5)} - \mu_3^{(3)} \mu_3^{(3)}$, $a_3^{(3)} = 1 - \mu_3^{(3)} + \left(\mu_3^{(3)}\right)^2$ |

2.3. Error estimates

In this paper, the well-known determination coefficient $R^2$ and the error $e$ are calculated respectively to measure the accuracy of the surrogate model, i.e.,

$$R^2 = 1 - \frac{\varepsilon_E}{\text{Var} \left( y_{\text{test}} \right)}$$

$$E \left( y_{\text{test}} \right) = \frac{1}{N_{\text{test}}} \sum_{l=1}^{N_{\text{test}}} y_{l \text{test}}$$

$$\text{Var} \left( y_{\text{test}} \right) = \frac{1}{N_{\text{test}} - 1} \sum_{l=1}^{N_{\text{test}}} \left[ y_{l \text{test}} - E \left( y_{\text{test}} \right) \right]^2$$

$$\varepsilon_E = \frac{1}{N_{\text{test}}} \sum_{l=1}^{N_{\text{test}}} \left[ y_{l \text{test}} - y_{l \text{test}} \left( \xi_{l \text{test}} \right) \right] \sqrt{\frac{\sum_{l=1}^{N_{\text{test}}} \left[ y_{l \text{test}} - y_{l \text{test}} \left( \xi_{l \text{test}} \right) \right]^2}{\sum_{l=1}^{N_{\text{test}}} \left( y_{l \text{test}} \right)^2}}$$

where $\xi_{l \text{test}}$ ($l = 1, 2, \cdots, N_{\text{test}}$) are the random inputs of test data, $y_{l \text{test}}$ are the corresponding true outputs of $\xi_{l \text{test}}$, $N_{\text{test}}$ is the number of test data, and $y_{l \text{test}}$ ($\xi_{l \text{test}}$) are the corresponding estimated outputs of $\xi_{l \text{test}}$ by the surrogate model. The situation $R^2 = 1$ or $e = 0$ corresponds to a perfect fit. Thus, $R^2 \to 1$ or $e \to 0$ means that the surrogate model can fit the stochastic model perfectly.

3. Deep arbitrary polynomial chaos expansion

3.1. Deep aPCE method framework

**Definition.** Defined that a function $F \left( \cdot \right)$ to calculate the expansion coefficients $C \left( \xi; \theta \right) = \{C_i \left( \xi; \theta \right) \mid i = 1, 2, \cdots, M \}$ for $\xi \in \Omega^d$, i.e.,

$$\Omega^d \longrightarrow \Omega^M : C \left( \xi; \theta \right) = F \left( \xi; \theta \right),$$

(23)
where $\theta$ is the parameter of function $F(\cdot)$. Based on the expansion coefficients $C(\xi; \theta)$, the aPC model of the stochastic system $Y = f(\xi) \ (\forall \xi \in \Omega^d)$ can be

$$
\hat{y}^{(p)}(\xi) = \sum_{i=1}^{M} C_i(\xi; \theta) \Phi_i(\xi).
$$

Supposed that $\theta^*$ is the optimal parameter. For $\theta \to \theta^*$, if the expansion coefficients $C(\xi; \theta)$ meet the following two conditions, i.e.,

$$
\left| \sum_{i=1}^{M} C_i(\xi; \theta) \Phi_i(\xi) - Y \right| < \varepsilon_1, \quad (\forall \varepsilon_1 > 0) \tag{25}
$$

$$
|C_i(\xi; \theta) - E[C_i(\xi; \theta)]| < \varepsilon_2, \quad (\forall \varepsilon_2 > 0; \ i = 1, 2, \cdots, M.), \tag{26}
$$

the coefficients $C(\xi; \theta) = \{C_i(\xi; \theta) \mid i = 1, 2, \cdots, M\}$ are called the adaptive expansion coefficients of the aPC model, and Eq.(24) is called the adaptive aPC model of the stochastic system $Y = f(\xi)$.

**Property.** Based on the orthonormality of the multi-dimensional orthogonal basis $\{\Phi_1(\xi), \Phi_2(\xi), \cdots, \Phi_M(\xi)\}$, the adaptive aPC model in Eq.(24) satisfies the following two properties for $\theta \to \theta^*$, i.e.,

$$
\left| E\left[\hat{y}^{(p)}(\xi)\right] - E[C_1(\xi; \theta)]\right| < \varepsilon_3, \quad (\forall \varepsilon_3 > 0), \tag{27}
$$

$$
\left| \text{Var}\left[\hat{y}^{(p)}(\xi)\right] - \sum_{i=2}^{M} \left\{E[C_i(\xi; \theta)]\right\} \right|^2 < \varepsilon_4, \quad (\forall \varepsilon_4 > 0), \tag{28}
$$

and the detailed proof process are shown in Appendix A.

Refer to the condition in Eqs. (23) and (26), the adaptive expansion coefficients are fine-tuned dynamically by the function $C(\xi; \theta) = F(\xi; \theta)$ for different random input variables $\xi$. For example, given random inputs $\xi^\text{test}_l \ (l = 1, 2, \cdots, n^\text{test})$, the adaptive expansion coefficients $C(\xi^\text{test}_l; \theta) = \{C_i(\xi^\text{test}_l; \theta) \mid i = 1, 2, \cdots, M\}$ will be determined by the function $C(\xi^\text{test}_l; \theta) = F(\xi^\text{test}_l; \theta)$ as shown in Fig.2, based on which the aPC model in Eq.(24) can accurately predict the corresponding output of the stochastic model $Y = f(\xi^\text{test})$. However, the problem is how to construct the function $C(\xi; \theta) = F(\xi; \theta)$ to make the coefficients $C(\xi; \theta) = \{C_i(\xi; \theta) \mid i = 1, 2, \cdots, M\}$ satisfy the above two conditions in Eqs.(25) and (26) for $\theta \to \theta^*$. According to the researches by Hornik et al. [33] and Cybenko [34], MLP is an universal approximator which can achieve any desired degree of accuracy by choosing a suitable MLP structure. Thus, the MLP model is used to construct the function $C(\xi; \theta) = F(\xi; \theta)$ in this paper.

Based on the above analysis, a Deep aPCE method is proposed to quantify the uncertainty of stochastic model $Y = f(\xi)$ based on DL and aPC, and the framework of Deep aPCE method is shown in Fig.3. Refer to Fig.3, the random input data $\{x \in \Omega^d\}_N$ are normalized firstly, i.e.,

$$
\xi = \frac{x - \mu_x}{\sigma_x}, \tag{29}
$$

where $\mu_x$ and $\sigma_x$ are the mean and the standard deviation of the random input data $x$, respectively. On the one hand, the moments of normalized input data are calculated by Eq.(17) and then the univariate orthogonal basis $\left\{\phi_k^{(0)}(\xi_k), \phi_k^{(1)}(\xi_k), \cdots, \phi_k^{(p)}(\xi_k)\right\}$ for random variables $\xi_k \ (k = 1, 2, \cdots, d)$ is constructed by Eq.(15). Thereby, the multi-dimensional orthogonal basis $\{\Phi_1(\xi), \Phi_2(\xi), \cdots, \Phi_M(\xi)\}$ can be obtained by Eq.(13)
based on the univariate orthogonal basis \( \{ \phi_k^{(0)}(\xi_k), \phi_k^{(1)}(\xi_k), \ldots, \phi_k^{(p)}(\xi_k) \} \). On the other hand, a suitable MLP model \( \mathcal{NN}(\xi; \theta) \) is constructed to solve the adaptive expansion coefficients \( c(\xi; \theta) \), i.e.,

\[
c(\xi; \theta) = \mathcal{NN}(\xi; \theta) = \{ c_1(\xi; \theta), c_2(\xi; \theta), \ldots, c_M(\xi; \theta) \}.
\]

(30)

According to Eq.(12), a Deep aPCE model \( \mathcal{DAPC}(\xi; \theta) \) can be built based on the multi-dimensional orthogonal basis \( \{ \Phi_1(\xi), \Phi_2(\xi), \ldots, \Phi_M(\xi) \} \) and the adaptive expansion coefficients \( c(\xi; \theta) \). Finally, based on the cost function \( J(\theta) \) and two kinds of training data sets as discussed in section 3.3, the parameters \( \theta = (W, b) \) of the Deep aPCE model \( \mathcal{DAPC}(\xi; \theta) \) are learned iteratively by the Adam algorithm [40]. After obtaining the learned parameters \( \theta = (W, b) \), the MLP model \( \mathcal{NN}(\xi; \theta) \) will meet the condition in Eq.(25). Therefore, the MLP model \( \mathcal{NN}(\xi; \theta) \) can be used to solve the adaptive expansion coefficients \( c(\xi; \theta) \), and the detailed proof process is shown in Appendix B.

Figure 3: The framework of the Deep aPCE method

In summary, based on the normalized input data, the constructing of Deep aPCE model \( \mathcal{DAPC}(\xi; \theta) \) main
includes the following four parts:

- Constructing the multi-dimensional orthogonal basis \( \{ \Phi_1(\xi), \Phi_2(\xi), \ldots, \Phi_M(\xi) \} \);
- Building a suitable MLP model \( \mathcal{N}(\xi; \theta) \) for solving the adaptive expansion coefficients \( c(\xi; \theta) \);
- Constructing a cost function \( \mathcal{J}(\theta) \) for training the Deep aPCE model \( \mathcal{DAPC}(\xi; \theta) \);
- Training the Deep aPCE model \( \mathcal{DAPC}(\xi; \theta) \) to learn its parameters \( \theta = (W, b) \).

3.2. Deep aPCE modeling

3.2.1. Constructing orthogonal basis

According to section 2.2, the univariate orthogonal basis \( \{ \phi_k^{(0)}(\xi_k), \phi_k^{(1)}(\xi_k), \ldots, \phi_k^{(p)}(\xi_k) \} \) is constructed by the raw moments of the random variable \( \xi_k \). Then, the multi-dimensional orthogonal basis

\[
\{ \Phi_1(\xi), \Phi_2(\xi), \ldots, \Phi_M(\xi) \}
\]

can be obtained by Eq.(13). In this section, a simple method is proposed for determining the multivariate index set \( s_i \) of the multi-dimensional polynomial \( \Phi_i(\xi) \) in Eq.(13).

Supposed that a \( p \)-order aPC model needs to be modeled for stochastic model \( Y = f(\xi) \) with the random input variable \( \xi = \{ \xi_k | k = 1, 2, \ldots, d \} \). Therefore, the number \( Q \) of order combinations of all univariates \( \xi_k \) is

\[
Q = (p + 1)^d. \tag{31}
\]

For \( q = 1, 2, \ldots, Q \), the \( k \)th index \( \tilde{s}_q^k \) of the multivariate index set \( \tilde{s}_q \) is determined by

\[
\tilde{s}_q^k = \begin{cases} 
(q - 1) \mod (p + 1) & k = d \\
\frac{q - 1}{(p + 1)^{d-k}} \mod (p + 1) & k = 1, 2, \ldots, (d - 1),
\end{cases} \tag{32}
\]

where \( x \mod y \) returns the remainder after division of \( x \) by \( y \), and \( \lfloor x \rfloor \) rounds \( x \) to the nearest integer less than or equal to \( x \). Thus, the multivariate index set \( \tilde{s}_q \) can be obtained, i.e.

\[
\tilde{s}_q = \{ \tilde{s}_1^q, \tilde{s}_2^q, \ldots, \tilde{s}_k^q, \ldots, \tilde{s}_d^q \}. \tag{33}
\]

Refer to Eq.(13), the sum of multivariate index \( \tilde{s}_q^k \) in \( \tilde{s}_q \) should less than or equal to \( p \), i.e.

\[
\sum_{k=1}^{d} \tilde{s}_q^k \leq p. \tag{34}
\]

Therefore, only \( M \) multivariate index sets meet the above conditions. Supposed that \( M \) multivariate index sets are \( \{ s_i | i = 1, 2, \ldots, M \} \), where \( s_i = \{ s_i^k | k = 1, 2, \ldots, d \} \).

For example, a two-dimensional orthogonal basis (2-order) is constructed in the following. Based on Eq.(15) and Table 1, two univariate orthogonal bases (2-order) are

\[
\begin{align*}
\phi_1^{(0)}(\xi_1) &= 1 & \phi_1^{(1)}(\xi_1) &= \xi_1 & \phi_1^{(2)}(\xi_1) &= (\xi_1)^2 - \mu_{\xi_1}^{(3)} \xi_1 - 1 \\
\phi_2^{(0)}(\xi_2) &= 1 & \phi_2^{(1)}(\xi_2) &= \xi_2 & \phi_2^{(2)}(\xi_2) &= (\xi_2)^2 - \mu_{\xi_2}^{(3)} \xi_2 - 1.
\end{align*}
\]
According to Eqs. (32) and (34), there are six two-variate index sets, i.e.
\[
\begin{align*}
  s_1 &= \{0,0\}, \quad s_2 = \{0,1\}, \quad s_3 = \{0,2\}, \\
  s_4 &= \{1,0\}, \quad s_5 = \{1,1\}, \quad s_6 = \{2,0\}.
\end{align*}
\]
Thus, based on the above two results, the two-dimensional orthogonal basis (2-order) can be obtained by Eq.(13), i.e.
\[
\begin{align*}
  \Phi_1 (\xi_1, \xi_2) &= 1, \\
  \Phi_2 (\xi_1, \xi_2) &= \xi_2, \\
  \Phi_3 (\xi_1, \xi_2) &= (\xi_2)^2 - \mu_{\xi_2}^2 - 1, \\
  \Phi_4 (\xi_1, \xi_2) &= \xi_1, \\
  \Phi_5 (\xi_1, \xi_2) &= \xi_1 \xi_2, \\
  \Phi_6 (\xi_1, \xi_2) &= (\xi_1)^2 - \mu_{\xi_1}^2 - 1.
\end{align*}
\]

3.2.2. Constructing MLP model

According to section 3.1, the adaptive expansion coefficients \(c (\xi; \theta)\) of the Deep aPCE model \(\text{DAPC} (\xi; \theta)\) are solved by the MLP model \(\mathcal{N} \mathcal{N} (\xi; \theta)\). In this paper, a \((L+2)\) layers MLP model \(\mathcal{N} \mathcal{N} (\xi; \theta)\) with the random input \(\xi = \{\xi_k | k = 1, 2, \cdots, d\}\) is constructed as shown in Fig.4. Due to the Deep aPCE model \(\text{DAPC} (\xi; \theta)\) has \(M\) expansion coefficients, the output layer has \(M\) neurons. As shown in Fig.4, there are \(L\) hidden layers. For the \(\alpha\)th \((\alpha = 1, 2, \cdots, L)\) hidden layer, it has \(n_\alpha\), neurons. Empirically, the number \(n_1\) of neurons in the first hidden layer is \(2.5\) to \(16\) times that of \(d\), and the number \(n_L\) of neurons in the last hidden layer is \(1\) to \(10\) times that of \(M\), i.e.,
\[
\begin{align*}
  n_1 &= (2.5 \sim 16) d, \\
  n_L &= (1 \sim 10) M.
\end{align*}
\]
It is noted that both \(n_1\) and \(n_L\) are integer. For the \(\alpha\)th \((\alpha = 2, \cdots, (L-1))\) hidden layer, the number \(n_\alpha\) of its neurons is generally greater than \(n_1\) or \(n_L\). Besides, ReLU \((x) [47, 48]\) is chosen to be the nonlinear activation function for each hidden layers’ neurons in the MLP model \(\mathcal{N} \mathcal{N} (\xi; \theta)\). For the MLP model \(\mathcal{N} \mathcal{N} (\xi; \theta)\), its parameters \(\theta\) can be learned based on the training data set, the cost function, and the deep neural network optimization algorithm. The detailed learning method will be discussed in section 3.4.

![MLP model for building the adaptive expansion coefficients](image)

3.2.3. Deep aPCE model

To quantify the uncertainty for stochastic model \(Y = f (\xi)\), a \(p\)-order Deep aPCE model \(\text{DAPC} (\xi; \theta)\) is constructed based on the multi-dimensional orthogonal basis \(\{\Phi_1 (\xi), \Phi_2 (\xi), \cdots, \Phi_M (\xi)\}\) and the adaptive expansion coefficients \(c (\xi; \theta)\), i.e.,
\[
\hat{g}^{(p)} (\xi; \theta) = \sum_{i=1}^{M} c_i (\xi; \theta) \Phi_i (\xi).
\]
The detailed schematic diagram is shown in Fig.5. For the Deep aPCE model $\mathcal{DAPC} (\xi; \theta)$, only the parameters $\theta$ need to be learned. If the parameters $\theta$ is determined, given the input $\xi = \{\xi_k | k = 1, 2, \cdots, d\}$, the corresponding output $\hat{y}^{(p)} (\xi; \theta)$ can be estimated by the Deep aPCE model $\mathcal{DAPC} (\xi; \theta)$ as shown in Fig.5. In order to make $\mathcal{DAPC} (\xi; \theta)$ approximate $Y = f (\xi)$ as accurately as possible, a cost function will be constructed in the following section. It is noteworthy that the final solved adaptive expansion coefficients are fine-tuned dynamically by the MLP model $NN (\xi; \theta)$ for different random input variables $\xi$. This is different from the existed PCE methods [12] in which expansion coefficients are constants.

![Figure 5: The schematic diagram of Deep aPCE model $\mathcal{DAPC} (\xi; \theta)$](image)

### 3.3. Cost function and its gradient

#### 3.3.1. Constructing cost function

In this section, a cost function $\mathcal{J} (\theta)$ is constructed to learn the parameters $\theta = (W, b)$ of the Deep aPCE model $\mathcal{DAPC} (\xi; \theta)$. To reduce the number of calling to expensive models such as finite element model, two kinds of training data sets are used to constructed the cost function $\mathcal{J} (\theta)$, i.e.,

- The labeled training data set $\mathcal{D}_{gd}$ with $N_{gd}$ data pairs $(x^{wd}_i, y^{wd}_i)$, i.e., $\mathcal{D}_{gd} = \{ (x^{wd}_i, y^{wd}_i) | l = 1, 2, \cdots, N_{gd} \}$. Apparently, each input $x^{wd}_i$ has a corresponding ground truth value $y^{wd}_i$.

- The non-labeled training data set $\mathcal{D}_{ce}$ with $N_{ce}$ inputs $x^{ce}_i$, i.e., $\mathcal{D}_{ce} = \{ x^{ce}_i | l' = 1, 2, \cdots, N_{ce} \}$. Different with the labeled training data set $\mathcal{D}_{gd}$, each data of the non-labeled training data set $\mathcal{D}_{ce}$ only has the input $x^{ce}_i$.

For the labeled training data set $\mathcal{D}_{gd}$, each input $x^{wd}_i$ is normalized firstly, i.e.,

$$\xi^{wd}_i = \frac{x^{wd}_i - \mu_{gd}}{\sigma_{gd}},$$  \hspace{1cm} (37)$$

where $\mu_{gd}$ and $\sigma_{gd}$ are the mean and the standard deviation of all inputs $x^{wd}_i \ (l = 1, 2, \cdots, N_{gd})$, respectively. For $l = 1, 2, \cdots, N_{gd}$, the normalized input $\xi^{wd}_i$ is input into the Deep aPCE model $\mathcal{DAPC} (\xi; \theta)$, and then the corresponding estimated value $\hat{y}^{wd}_i (\xi^{wd}_i; \theta)$ can be obtained, i.e.,

$$\hat{y}^{wd}_i (\xi^{wd}_i; \theta) = \sum_{i=1}^{M} c_i (\xi^{wd}_i; \theta) \Phi_i (\xi^{wd}_i).$$  \hspace{1cm} (38)$$
Thus, the mean absolute error $L_{gd}(\xi_{l}^{sd}, y_{l}^{sd}; \theta)$ between the estimated values $y_{l}^{sd}(\xi_{l}^{sd}; \theta)$ and the ground truth values $y_{l}^{sd}$ ($l = 1, 2, \ldots, N_{gd}$) is

$$L_{gd}(\xi_{l}^{sd}, y_{l}^{sd}; \theta) = \frac{1}{N_{gd}} \sum_{l=1}^{N_{gd}} |\hat{y}_{l}^{sd}(\xi_{l}^{sd}; \theta) - y_{l}^{sd}|$$

For the non-labeled training data set $\mathcal{D}_{ce}$, each input $x_{l}^{ce}$ is normalized firstly, i.e.,

$$\xi_{l}^{ce} = \frac{x_{l}^{ce} - \mu_{ce}}{\sigma_{ce}},$$

where $\mu_{ce}$ and $\sigma_{ce}$ are the mean and the standard deviation of all inputs $x_{l}^{ce}$ ($l' = 1, 2, \ldots, N_{ce}$), respectively. For $l' = 1, 2, \ldots, N_{ce}$, the normalized input $\xi_{l'}^{ce}$ is input into the Deep aPCE model $\mathcal{DAPC}(\xi; \theta)$, and then $N_{ce}$ sets of expansion coefficients $c(\xi_{l'}^{ce}; \theta)$ are

$$c(\xi_{l'}^{ce}; \theta) = \{c_{1}(\xi_{l'}^{ce}; \theta), c_{2}(\xi_{l'}^{ce}; \theta), \ldots, c_{M}(\xi_{l'}^{ce}; \theta)\}$$

Therefore, the means of expansion coefficients $c_{i}(\xi_{l'}^{ce}; \theta)$ ($i = 1, 2, \ldots, M$) are

$$E[c_{1}(\xi_{l'}^{ce}; \theta)] = \frac{1}{N_{ce}} \sum_{l=1}^{N_{ce}} c_{1}(\xi_{l'}^{ce}; \theta)$$

$$E[c_{2}(\xi_{l'}^{ce}; \theta)] = \frac{1}{N_{ce}} \sum_{l=1}^{N_{ce}} c_{2}(\xi_{l'}^{ce}; \theta)$$

$$\vdots$$

$$E[c_{i}(\xi_{l'}^{ce}; \theta)] = \frac{1}{N_{ce}} \sum_{l=1}^{N_{ce}} c_{i}(\xi_{l'}^{ce}; \theta)$$

$$E[c_{M}(\xi_{l'}^{ce}; \theta)] = \frac{1}{N_{ce}} \sum_{l=1}^{N_{ce}} c_{M}(\xi_{l'}^{ce}; \theta).$$

In addition to expansion coefficients $c(\xi_{l'}^{ce}; \theta)$, the corresponding estimated values $\hat{y}_{l'}^{ce}(\xi_{l'}^{ce}; \theta)$ ($l' = 1, 2, \ldots, N_{ce}$) for the normalized input $\xi_{l'}^{ce}$ can be estimated by the Deep aPCE model $\mathcal{DAPC}(\xi; \theta)$, i.e.,

$$\hat{y}_{l'}^{ce}(\xi_{l'}^{ce}; \theta) = \sum_{i=1}^{M} c_{i}(\xi_{l'}^{ce}; \theta) \Phi_{i}(\xi_{l'}^{ce}).$$

Thus, the mean and the variance of estimated values $\hat{y}_{l'}^{ce}(\xi_{l'}^{ce}; \theta)$ ($l' = 1, 2, \ldots, N_{ce}$) are calculated respectively by

$$E[\hat{y}_{l'}^{ce}(\xi_{l'}^{ce}; \theta)] = \frac{1}{N_{ce}} \sum_{l=1}^{N_{ce}} \hat{y}_{l'}^{ce}(\xi_{l'}^{ce}; \theta)$$

$$= \frac{1}{N_{ce}} \sum_{l=1}^{N_{ce}} \left[ \sum_{i=1}^{M} c_{i}(\xi_{l'}^{ce}; \theta) \Phi_{i}(\xi_{l'}^{ce}) \right].$$

$$= \frac{1}{N_{ce}} \sum_{l=1}^{N_{ce}} \left[ \sum_{i=1}^{M} c_{i}(\xi_{l'}^{ce}; \theta) \Phi_{i}(\xi_{l'}^{ce}) \right].$$
Besides, the adaptive expansion coefficients are fine-tuned dynamically by the MLP model for different random input variables $\xi$. Thus, the variance of the adaptive expansion coefficients need to be as small as possible, i.e.,

$$
\mathcal{L}_{cc}^{\text{var}} (\xi^{cc}; \theta) = \sum_{i=1}^{M} \left\{ \frac{1}{N_{ce} - i} \sum_{l'=1}^{N_{ce}} \left[ c_i (\xi^{cc}_{l'}; \theta) - E [c_i (\xi^{cc}_{l'}; \theta)] \right]^2 \right\}
$$

$$
= \frac{1}{N_{ce} - i} \sum_{l'=1}^{N_{ce}} \left[ \sum_{i=1}^{M} c_i (\xi^{cc}_{l'}; \theta) \Phi_i (\xi^{cc}_{l'}) - \frac{1}{N_{ce} \sum_{l'=1}^{M}} \left[ \sum_{i=1}^{M} c_i (\xi^{cc}_{l'}; \theta) \Phi_i (\xi^{cc}_{l'}) \right] \right]^2.
$$

(48)

Therefore, the absolute error $\mathcal{L}_{cc} (\xi^{cc}_{l'}; \theta)$ of the non-labeled training data set $\mathcal{D}_{ce}$ is

$$
\mathcal{L}_{cc} (\xi^{cc}_{l'}; \theta) = \mathcal{L}_{cc}^{1} (\xi^{cc}_{l'}; \theta) + \mathcal{L}_{cc}^{2M} (\xi^{cc}_{l'}; \theta) + \mathcal{L}_{cc}^{\text{var}} (\xi^{cc}_{l'}; \theta).
$$

(49)
In summary, the cost function $J(\theta)$ for learning the parameters $\theta$ of the Deep aPCE model $\mathcal{DAPC}(\xi; \theta)$ is
\[
J(\theta) = L_{gd} \left( \xi_i^{gd}, y_i^{gd}; \theta \right) + L_{ce} \left( \xi_i^{ce}; \theta \right) \\
= L_{gd} \left( \xi_i^{gd}, y_i^{gd}; \theta \right) + L_{ce}^1 \left( \xi_i^{ce}; \theta \right) + L_{ce}^{2m} \left( \xi_i^{ce}; \theta \right) + L_{ce}^{var} \left( \xi_i^{ce}; \theta \right).
\]
(50)

As shown in Fig.6, supposed that parameters $\theta^* = (W^*, b^*)$ are the optimal parameters. By minimizing the proposed cost function $J(\theta)$ in Eq.(50), the parameters $\theta = (W, b)$ can be learned to approach the optimal parameters $\theta^*$, i.e.,
\[
\theta^* \leftarrow \theta = \arg \min_{\theta} J(\theta) = \arg \min_{\theta} \left[ L_{gd} \left( \xi_i^{gd}, y_i^{gd}; \theta \right) + L_{ce}^1 \left( \xi_i^{ce}; \theta \right) + L_{ce}^{2m} \left( \xi_i^{ce}; \theta \right) + L_{ce}^{var} \left( \xi_i^{ce}; \theta \right) \right].
\]
(51)

![Figure 6: Schematic diagram of parameters $\theta = (W, b)$ approaching the optimal parameters $\theta^* = (W^*, b^*)$](image)

According to Appendix A, the necessary condition for Eqs.(27) and (28) is that the $i$th ($i = 1, 2, \cdots, M$) adaptive expansion coefficient $C_i (\xi)$ approaches its expectation $E [C_i (\xi)]$ for $\theta \rightarrow \theta^*$. This is equivalent to the variance of each adaptive expansion coefficient approaches zero for $\theta \rightarrow \theta^*$. Thus, while minimizing $L_{ce}^1 (\xi_i^{ce}; \theta)$ and $L_{ce}^{2m} (\xi_i^{ce}; \theta)$, the variance of each adaptive expansion coefficient is also minimized implicitly. Besides, combined with the results of numerical examples, if the last part $L_{ce}^{var} (\xi_i^{ce}; \theta)$ of the cost function $J(\theta)$ is removed in training the Deep aPCE model, the trained Deep aPCE model still can accurately approximate the stochastic system. Thus, the cost function $J(\theta)$ in Eq.(50) can be simplified to
\[
J(\theta) = L_{gd} \left( \xi_i^{gd}, y_i^{gd}; \theta \right) + L_{ce}^1 \left( \xi_i^{ce}; \theta \right) + L_{ce}^{2m} \left( \xi_i^{ce}; \theta \right) + L_{ce}^{var} \left( \xi_i^{ce}; \theta \right).
\]
(52)

By minimizing the simplified cost function $J(\theta)$ in Eq.(52), the parameters $\theta = (W, b)$ can also be learned to approach the optimal parameters $\theta^*$, i.e.,
\[
\theta^* \leftarrow \theta = \arg \min_{\theta} J(\theta) = \arg \min_{\theta} \left[ L_{gd} \left( \xi_i^{gd}, y_i^{gd}; \theta \right) + L_{ce}^1 \left( \xi_i^{ce}; \theta \right) + L_{ce}^{2m} \left( \xi_i^{ce}; \theta \right) \right].
\]
(53)

3.3.2. Gradient of cost function

According to section 2.1, the parameters $\theta = (W, b)$ of the Deep aPCE model $\mathcal{DAPC}(\xi; \theta)$ are learned based on the gradient $\mathcal{G}(\theta)$ of the cost function $J(\theta)$. Therefore, this section will introduce how to solve the gradient $\mathcal{G}(\theta)$. Refer to Eq.(50), the cost function $J(\theta)$ includes four parts: $L_{gd} \left( \xi_i^{gd}, y_i^{gd}; \theta \right), L_{ce}^1 \left( \xi_i^{ce}; \theta \right), L_{ce}^{2m} \left( \xi_i^{ce}; \theta \right)$, and $L_{ce}^{var} \left( \xi_i^{ce}; \theta \right)$.

For the mean absolute error $L_{gd} \left( \xi_i^{gd}, y_i^{gd}; \theta \right)$ in Eq.(39), its partial derivative with respect to parameters $\theta$ is
\[
\frac{\partial L_{gd} \left( \xi_i^{gd}, y_i^{gd}; \theta \right)}{\partial \theta} = \frac{1}{N_{gd}} \sum_{t=1}^{N_{gd}} \left[ \gamma_t \sum_{i=1}^{M} \Phi_i \left( \xi_i^{gd} \right) \frac{\partial \gamma_i \left( \xi_i^{gd}; \theta \right)}{\partial \theta} \right],
\]
(54)
where γl is an intermediate variable. If \( \hat{y}^d_l (\xi^{rd}_l; \theta) > y^d_l \), γl = 1, and if \( \hat{y}^d_l (\xi^{rd}_l; \theta) < y^d_l \), γl = -1, otherwise γl = 0.

For the absolute error \( \mathcal{L}_{ce}^1 (\xi^{ce}; \theta) \) in Eq.(46), its partial derivative with respect to parameters \( \theta \) is

\[
\frac{\partial \mathcal{L}_{ce}^1 (\xi^{ce}; \theta)}{\partial \theta} = \frac{\beta_1}{N_{ce}} \sum_{i=1}^{N_{ce}} \left[ \sum_{l=1}^{M} \Phi_l (\xi^{ce}_l; \theta) \frac{\partial c_i (\xi^{ce}_l; \theta)}{\partial \theta} - \frac{\partial c_1 (\xi^{ce}_1; \theta)}{\partial \theta} \right],
\]

where \( \beta_1 \) is an intermediate variable. If \( E[\hat{y}^{ce}_l (\xi^{ce}; \theta)] > E[c_1 (\xi^{ce}; \theta)] \), \( \beta_1 = 1 \), and if \( E[\hat{y}^{ce}_l (\xi^{ce}; \theta)] < E[c_1 (\xi^{ce}; \theta)] \), \( \beta_1 = -1 \), otherwise \( \beta_1 = 0 \).

For the absolute error \( \mathcal{L}_{ce}^2M (\xi^{ce}_l; \theta) \) in Eq.(47), its partial derivative with respect to parameters \( \theta \) is

\[
\frac{\partial \mathcal{L}_{ce}^2M (\xi^{ce}_l; \theta)}{\partial \theta} = \frac{2\beta_2}{N_{ce} - 1} \sum_{i=1}^{M} \sum_{t=1}^{N_{ce}} \left[ \sum_{l=1}^{M} \mathcal{R} \Phi_l (\xi^{ce}_t; \theta) \frac{\partial c_i (\xi^{ce}_t; \theta)}{\partial \theta} - \frac{\mathcal{R} c_i (\xi^{ce}_1; \theta)}{N_{ce}} \sum_{l=1}^{M} \Phi_l (\xi^{ce}_l; \theta) \right]
\]

\[
- \frac{2\beta_2}{N_{ce} - 1} \sum_{i=1}^{M} \sum_{t=1}^{N_{ce}} \left[ E[c_i (\xi^{ce}; \theta)] \sum_{l=1}^{M} \frac{\partial c_i (\xi^{ce}_l; \theta)}{\partial \theta} \right],
\]

where

\[
\mathcal{R} = \sum_{i=1}^{M} c_i (\xi^{ce}_l; \theta) \Phi_l (\xi^{ce}_t; \theta) - \frac{1}{N_{ce}} \sum_{l=1}^{M} \sum_{t=1}^{N_{ce}} \Phi_l (\xi^{ce}_l; \theta) \Phi_l (\xi^{ce}_t; \theta),
\]

and \( \beta_2 \) is an intermediate variable. If \( \text{Var}[\hat{y}^{ce}_l (\xi^{ce}; \theta)] > \sum_{i=2}^{M} \{ E[c_i (\xi^{ce}; \theta)] \}^2 \), \( \beta_2 = 1 \), and if \( \text{Var}[\hat{y}^{ce}_l (\xi^{ce}; \theta)] < \sum_{i=2}^{M} \{ E[c_i (\xi^{ce}; \theta)] \}^2 \), \( \beta_2 = -1 \), otherwise \( \beta_2 = 0 \).

For \( \mathcal{L}_{var}^1 (\xi^{ce}_l; \theta) \) in Eq.(48), its partial derivative with respect to parameters \( \theta \) is

\[
\frac{\partial \mathcal{L}_{var}^1 (\xi^{ce}_l; \theta)}{\partial \theta} = \frac{2}{N_{ce} - 1} \sum_{i=1}^{M} \sum_{t=1}^{N_{ce}} \left[ c_i (\xi^{ce}_l; \theta) - \frac{1}{N_{ce}} \sum_{t=1}^{N_{ce}} c_i (\xi^{ce}_t; \theta) \right] \left[ \frac{\partial c_i (\xi^{ce}_l; \theta)}{\partial \theta} - \frac{1}{N_{ce}} \sum_{t=1}^{N_{ce}} \frac{\partial c_1 (\xi^{ce}_t; \theta)}{\partial \theta} \right]
\]

In Eqs.(54), (55), (56) and (58), only \( \frac{\partial c_i (\xi^{ce}_l; \theta)}{\partial \theta} \) or \( \frac{\partial c_1 (\xi^{ce}_l; \theta)}{\partial \theta} \) needs to be solved. In DL, \( \frac{\partial c_i (\xi^{ce}_l; \theta)}{\partial \theta} \) and \( \frac{\partial c_1 (\xi^{ce}_l; \theta)}{\partial \theta} \) can be derived by the chain rule for differentiating compositions of functions using automatic differentiation [49]. Based on Eqs.(54), (55), (56) and (58), the gradient \( \mathcal{G}(\theta) \) of the cost function \( \mathcal{J}(\theta) \) can be obtained, i.e.,

\[
\mathcal{G}(\theta) = \frac{\partial \mathcal{J}(\theta)}{\partial \theta} = \frac{\partial \mathcal{L}_{gd} (\xi^{rd}_l; y^d_l; \theta)}{\partial \theta} + \frac{\partial \mathcal{L}_{ce}^1 (\xi^{ce}_l; \theta)}{\partial \theta} + \frac{\partial \mathcal{L}_{ce}^2M (\xi^{ce}_l; \theta)}{\partial \theta} + \frac{\partial \mathcal{L}_{var}^1 (\xi^{ce}_l; \theta)}{\partial \theta}.
\]

3.4. Deep aPCE model training

According to section 3.2.3, the Deep aPCE model \( \mathcal{DAPC}(\xi; \theta) \) needs to be trained for learning parameters \( \theta = (W, b) \). Therefore, this section proposes an iterative training method for training the Deep aPCE model \( \mathcal{DAPC}(\xi; \theta) \), and its flowchart is shown in Fig.7. Firstly, setting the maximum training epoch \( ep_{max} \), and the MLP model \( N\mathcal{N}(\xi; \theta) \) is initialized randomly so that the parameters \( \theta = (W, b) \) have initial values. Then, given the labeled training data set \( \mathcal{D}_{gd} \) and the non-labeled training data set \( \mathcal{D}_{ce} \), the corresponding expansion coefficients and the estimated values can be obtained by the Deep aPCE model \( \mathcal{DAPC}(\xi; \theta) \) in Fig.5. Subsequently, the cost function \( \mathcal{J}(\theta) \) of the Deep aPCE model \( \mathcal{DAPC}(\xi; \theta) \) is calculated by Eq.(50).

Therefore, the gradient \( \mathcal{G}(\theta) \) is obtained using the cost function \( \mathcal{J}(\theta) \) according to the gradient solving method in section 3.3.2. Refer to section 2.1, the parameters \( \theta = (W, b) \) are updated by Eq.(11). In this
In the paper, the Adam algorithm [40] is selected as the optimization solution algorithm. For the Adam algorithm, the corresponding calculation update operator $\Delta [\eta, \mathcal{G}(\theta)]$ is derived in the following.

For the $ep$-th training epoch, the biased first moment estimate is

$$s_{ep} = \rho_1 s_{ep-1} + (1 - \rho_1) \mathcal{G}(\theta),$$  \hspace{1cm} (60)

and the biased second raw moment estimate is

$$r_{ep} = \rho_2 r_{ep-1} + (1 - \rho_2) \mathcal{G}(\theta) \odot \mathcal{G}(\theta),$$  \hspace{1cm} (61)

where $\rho_1$ and $\rho_2$ ($\rho_1, \rho_2 \in [0, 1]$) are the exponential decay rates for the moment estimates, and $\odot$ is the Hadamard product. In particular, when $ep = 1$, $s_0 = r_0 = 0$. Then, the bias-corrected first moment estimate is computed by

$$\hat{s}_{ep} = \frac{s_{ep}}{1 - (\rho_1)^{ep}},$$  \hspace{1cm} (62)
and the bias-corrected second raw moment estimate is computed by

\[ \hat{r}_{ep} = \frac{r_{ep}}{1 - (\rho_2)_{ep}}. \]  

(63)

Thus, the calculation update operator \( \Delta [\eta, \mathcal{G}(\theta)] \) is

\[ \Delta [\eta, \mathcal{G}(\theta)] = -\eta \frac{\hat{s}_{ep}}{\sqrt{\hat{r}_{ep} + \delta}}, \]  

(64)

where \( \delta = 10^{-8} \).

Refer to Eq.(11) in section 2.1, the parameters \( \theta = (\mathbf{W}, \mathbf{b}) \) of the Deep aPCE model \( \mathcal{DAPC}(\xi; \theta) \) are updated iteratively until \( ep = ep_{\text{max}} \) as shown in Fig.7. In summary, the pseudo code for training the Deep aPCE model \( \mathcal{DAPC}(\xi; \theta) \), it can estimate the output corresponding to the new random input \( \mathbf{x} \). It is noteworthy that the new random input \( \mathbf{x} \) should be normalized firstly by Eq.(29).

3.5. Expansion order discussion

For a stochastic system with \( d \) random input variables, its \( p \)-order PCE model has \( M_p = (d + p)!/(d!p!) \) expansion coefficients. For \( p = 2, 3, 4, 5 \), the number of expansion coefficients are

\[
\begin{align*}
M_2 &= \frac{1}{2} (d^2 + 3d + 2) \\
M_3 &= M_2 + \frac{1}{3} M_2 d \\
M_4 &= M_2 + M_2 \left( \frac{1}{12} d^2 + \frac{7}{12} d \right) \\
M_5 &= M_2 + M_2 \left( \frac{1}{60} d^3 + \frac{1}{5} d^2 + \frac{47}{60} d \right).
\end{align*}
\]  

(65)

As shown in Eq.(65), \( M_5, M_4, \) and \( M_5 \) are all greater than \( M_2 \). Besides, as shown in Fig.8, the number of expansion coefficients increases with the increasing of the random input variable’s dimensionality \( d \), and the number of the 2-order PCE model’s expansion coefficients is much less than higher orders PCE model. Especially, the larger the expansion order, the more the number of expansion coefficients increases.

Figure 8: The relationship between the amount of expansion coefficients and the dimensions for different expansion orders. \( M_p \) \( (p = 2, 3, 4, 5) \) is the number of expansion coefficients for \( p \)-order PCE model, and \( M_p - M_2 \) \( (p = 3, 4, 5) \) is the value of \( M_p \) minus \( M_2 \).
Algorithm: Deep aPCE model $\mathcal{DAPC}(\xi; \theta)$ training algorithm.

Input:

1. Learning rate $\eta$;
2. Maximum training epoch $ep_{\text{max}}$;
3. Labeled training data set $\mathcal{D}_{gd} = \{(x_{gdl}, y_{gdl}) | l = 1, 2, \ldots, N_{gd}\}$;
4. Non-labeled training data set $\mathcal{D}_{ce} = \{(x_{cel}' | l' = 1, 2, \ldots, N_{ce}\}$.

Output:
Trained Deep aPCE model $\mathcal{DAPC}(\xi; \theta)$.

1. Initialize the MLP model $\mathcal{NN}(\xi; \theta)$;
2. Normalize the labeled training data set $\mathcal{D}_{gd}$ to get $\{(\xi_{gd}^l, y_{gd}^l) | l = 1, 2, \ldots, N_{gd}\}$;
3. Normalize the non-labeled training data set $\mathcal{D}_{ce}$ to get $\{\xi_{ce}^{l'} | l' = 1, 2, \ldots, N_{ce}\}$;
4. for $ep = 1 : ep_{\text{max}}$ do
5.  for $l = 1 : N_{gd}$ do
6.    Calculate the corresponding estimated value $\hat{y}_{gdl}^l(\xi_{gd}^l; \theta) = \mathcal{DAPC}(\xi_{gd}^l; \theta)$;
7.  end
8.  Calculate the mean absolute error $L_{gd}(\xi_{gd}^l, y_{gd}^l; \theta)$;
9.  for $l' = 1 : N_{ce}$ do
10.    Calculate the expansion coefficients $c(\xi_{ce}^{l'}; \theta)$;
11.    Calculate the corresponding estimated value $\hat{y}_{ce}^{l'}(\xi_{ce}^{l'}; \theta) = \mathcal{DAPC}(\xi_{ce}^{l'}; \theta)$;
12.  end
13.  for $i = 1 : M$ do
14.    Calculate the means $E[c_i(\xi_{ce}; \theta)]$ of expansion coefficient $c_i(\xi_{ce}; \theta)$;
15.  end
16.  Calculate the mean $E[\hat{y}_{ce}^{l'}(\xi_{ce}; \theta)]$ and variance $Var[\hat{y}_{ce}^{l'}(\xi_{ce}; \theta)]$ of estimated values $\hat{y}_{ce}^{l'}(\xi_{ce}; \theta)$ ($l' = 1, 2, \ldots, N_{ce}$);
17.  Calculate the absolute error $L_{ce}(\xi_{ce}^{l'}, \theta)$;
18.  Calculate the cost function $J(\theta) = L_{gd}(\xi_{gd}^l, y_{gd}^l; \theta) + L_{ce}(\xi_{ce}^{l'}; \theta)$;
19.  Calculate the gradient $G(\theta)$ of the cost function $J(\theta)$;
20.  Calculate the calculation update operator $\Delta[\eta, G(\theta)]$;
21.  Update the parameters $\theta \leftarrow \theta + \Delta[\eta, G(\theta)]$;
22. end

Generally, the expansion order needs to be increased to construct a more accurate PCE model for a complex stochastic system. However, this will cause more labeled data to solve the expansion coefficients. Thus, in the case of ensuring the accuracy of the PCE model, a lower expansion order can reduce the required amount of the labeled data. In this paper, the proposed Deep aPCE method can alleviate the above contradiction. On the one hand, the final solved adaptive expansion coefficients are fine-tuned dynamically by the MLP model $\mathcal{NN}(\xi; \theta)$
for different random input variables $\boldsymbol{\xi}$ according to section 3.2.3. On the other hand, in addition to the labeled training data, some non-labeled training data are used to solved the adaptive expansion coefficients refer to section 3.3.1. On the basis of the above two points, the expansion coefficients of Deep aPCE can be solved more accurately. Thus, a low-order expanded Deep aPCE model can accurately approximate a stochastic system. As shown in section 4, the 2-order Deep aPCE model $\mathcal{DAPC}(\boldsymbol{\xi}; \boldsymbol{\theta})$ can construct accurate surrogate model for the low, medium, and appropriate high dimensional stochastic systems.

4. Numerical examples

In this section, four numerical examples are used for verifying the effectiveness of the proposed Deep aPCE method. For each numerical example, the values of random input variables are sampled by Latin Hypercube Sampling, and a 2-order aPC model is used to construct the Deep aPCE model. Besides, the MLP in the Deep aPCE model is built by PyTorch$^1$. Especially, the solving process of the first numerical example will be introduced in detail to demonstrate the usage of the Deep aPCE method. The other three numerical examples’ solving processes are similar to the first numerical example. Besides, to validate the effectiveness of two absolute errors $L^1_{ce}(\xi_{ce}^l; \boldsymbol{\theta})$ and $L^{2M}_{ce}(\xi_{ce}^l; \boldsymbol{\theta})$ in training the Deep aPCE model, the second numerical example will compare the accuracies of two Deep aPCE models, which are obtained by the cost function $J(\boldsymbol{\theta})$ and the mean absolute error $L_{gd}(\xi_{gd}^{pd}; y_{gd}^{pd}; \boldsymbol{\theta})$, respectively. The relevant codes of the Deep aPCE method by Python are available on this website$^2$.

4.1. Example 1: Fortini’s clutch

The first example is the contact angle $y$ uncertainty analysis of Fortini’s clutch [50, 51, 52], as shown in Fig.9.

![Fortini's clutch](image)

Figure 9: Fortini’s clutch

The contact angle $y$ can be calculated by four independent normally distributed variables $X_1$, $X_2$, $X_3$ and $X_4$ as follows:

$$y = \arccos \left( \frac{X_1 + 0.5 (X_2 + X_3)}{X_4 - 0.5 (X_2 + X_3)} \right),$$  \hspace{1cm} (66)

---

$^1$https://pytorch.org/get-started/locally/

$^2$https://github.com/Xiaohu-Zheng/Deep-aPCE
where the distribution parameters of $X_1$, $X_2$, $X_3$ and $X_4$ are shown in Table 2.

| Variable | mean (mm) | Variance (mm$^2$) | Distribution |
|----------|-----------|------------------|--------------|
| $X_1$    | 55.29     | 0.0793$^2$       | Normal       |
| $X_2$    | 22.86     | 0.0043$^2$       | Normal       |
| $X_3$    | 22.86     | 0.0043$^2$       | Normal       |
| $X_4$    | 101.6     | 0.0793$^2$       | Normal       |

**Deep aPCE modeling.** For this example, the random input variable $\mathbf{X} = \{X_1, X_2, X_3, X_4\}$ is normalized to be $\mathbf{\xi} = \{\xi_1, \xi_2, \xi_3, \xi_4\}$. Then, for $k = 1, 2, 3, 4$, the first three raw moments are calculated by Eq.(17), i.e.,

$$
\mu_\xi^{(1)} = 0 \quad \mu_\xi^{(2)} = 1 \quad \mu_\xi^{(3)} = 0.
$$

Thus, four univariate orthogonal bases (2-order) are constructed by Eq.(15) and Table 1, i.e.,

$$
\phi_k^{(0)} (\xi_k) = 1 \quad \phi_k^{(1)} (\xi_k) = \xi_k \quad \phi_k^{(2)} (\xi_k) = \xi_k^2 - 1,
$$

where $k = 1, 2, 3, 4$. Besides, according to Eqs.(32) and (34), there are 15 multivariate index sets, i.e.,

- $s_1 = \{0, 0, 0, 0\}$  
- $s_2 = \{0, 0, 0, 1\}$  
- $s_3 = \{0, 0, 0, 2\}$  
- $s_4 = \{0, 0, 1, 0\}$  
- $s_5 = \{0, 0, 1, 1\}$  
- $s_6 = \{0, 0, 2, 0\}$  
- $s_7 = \{0, 1, 0, 0\}$  
- $s_8 = \{0, 1, 0, 1\}$  
- $s_9 = \{0, 1, 1, 0\}$  
- $s_{10} = \{0, 2, 0, 0\}$  
- $s_{11} = \{1, 0, 0, 0\}$  
- $s_{12} = \{1, 0, 0, 1\}$  
- $s_{13} = \{1, 0, 1, 0\}$  
- $s_{14} = \{1, 1, 0, 0\}$  
- $s_{15} = \{2, 0, 0, 0\}$  

Based on the above results, four-dimensional orthogonal basis can be obtained by Eq.(13), i.e.,

$$
\Phi_1 (\mathbf{\xi}) = \mathbf{1} \quad \Phi_2 (\mathbf{\xi}) = \mathbf{\xi}_1 \quad \Phi_3 (\mathbf{\xi}) = (\mathbf{\xi}_1)^2 - 1 \quad \Phi_4 (\mathbf{\xi}) = \mathbf{\xi}_3 \quad \Phi_5 (\mathbf{\xi}) = \mathbf{\xi}_3 \mathbf{\xi}_4
$$

$$
\Phi_6 (\mathbf{\xi}) = (\mathbf{\xi}_3)^2 - 1 \quad \Phi_7 (\mathbf{\xi}) = \mathbf{\xi}_2 \quad \Phi_8 (\mathbf{\xi}) = \mathbf{\xi}_2 \mathbf{\xi}_4 \quad \Phi_9 (\mathbf{\xi}) = \mathbf{\xi}_2 \mathbf{\xi}_3 \quad \Phi_{10} (\mathbf{\xi}) = (\mathbf{\xi}_2)^2 - 1
$$

$$
\Phi_{11} (\mathbf{\xi}) = \mathbf{\xi}_1 \quad \Phi_{12} (\mathbf{\xi}) = \mathbf{\xi}_1 \mathbf{\xi}_4 \quad \Phi_{13} (\mathbf{\xi}) = \mathbf{\xi}_1 \mathbf{\xi}_3 \quad \Phi_{14} (\mathbf{\xi}) = \mathbf{\xi}_1 \mathbf{\xi}_2 \quad \Phi_{15} (\mathbf{\xi}) = (\mathbf{\xi}_1)^2 - 1,
$$

where $\mathbf{\xi} = \{\xi_1, \xi_2, \xi_3, \xi_4\}$. Therefore, the 2-order Deep aPCE model $\mathcal{DAPC} (\mathbf{\xi}; \mathbf{\theta})$ is

$$
y^{(2)} (\mathbf{\xi}) = \sum_{i=1}^{15} c_i (\mathbf{\xi}; \mathbf{\theta}) \Phi_i (\mathbf{\xi}),
$$

where $c_i (\mathbf{\xi}; \mathbf{\theta})$ is calculated by the MLP model $\mathcal{N}\mathcal{V} (\mathbf{\xi}; \mathbf{\theta})$. In this example, an MLP model with seven layers is adopted as shown in Fig.10. As shown in Fig.10, this MLP model has 4 inputs and 15 outputs. Besides, the neuron numbers of 5 hidden layers are 64, 128, 256, 128, and 64, respectively.

**Preparing training data set.** There are two kinds of training data sets, i.e., the labeled training data set $\mathcal{D}_{gd} = \{ (\mathbf{X}_i^{gd}, y_i^{gd}) \mid l = 1, 2, \cdots, N_{gd} \}$ and the non-labeled training data set $\mathcal{D}_{ce} = \{ \mathbf{X}_l^{ce} \mid l = 1, 2, \cdots, N_{ce} \}$, where $y_i^{gd}$ is calculated by Eq.(66) given $\mathbf{X}_i^{gd}$. The inputs $\mathbf{X}_i^{gd}$ and $\mathbf{X}_l^{ce}$ are normalized to be $\mathbf{\xi}_i^{gd}$ and $\mathbf{\xi}_l^{ce}$ by Eq.(37) and Eq.(40), respectively. In this example, $10^5$ non-labeled training data make up data set $\mathcal{D}_{ce}$, i.e., $N_{ce} = 10^5$. Besides, the number $N_{gd}$ of labeled training data is 17, 30, and 40, respectively.
Deep aPCE model training. Based on the labeled training data set $D_{gd} = \{ (\xi_{ld}^{gd}, y_{ld}^{gd}) | l = 1, 2, \cdots, N_{gd} \}$ ($N_{gd} = 17, 30, 40$) and the non-labeled training data set $D_{ce} = \{ \xi_{ld}^{ce} | l' = 1, 2, \cdots, 10^5 \}$, the Deep aPCE model is trained. The maximum training epoch $ep_{max}$ is set to be 7000. Besides, the initial learning rate $\eta$ is set to be 0.01. During the model training process, the learning rate $\eta$ is scaled by 0.8 times every 300 epochs, i.e.,

$$\eta \leftarrow 0.8 \times \eta, \text{ if } (ep \mod 300) = 1.$$ 

Results analysis. Based on the trained Deep aPCE model $\mathcal{DAPC}(\xi; \theta)$, the uncertainty analysis results of this example are shown in Table 3. In Table 3, the results by the MCS method ($10^6$ runs) are regarded as the true results for comparison. Besides, to validate the effectiveness of Deep aPCE method, this example is also solved by 5$^n$ univariate dimension reduction method ($5^n$ UDR) [51], 2-order, 3-order, and 4-order PCE method [10], respectively. Compared with the results of the MCS method, $\varepsilon_{Pr}$ denote the estimation error rate of $Pr(y < 6 \text{deg})$, i.e.,

$$\varepsilon_{Pr} = \frac{|Pr(y < 6 \text{deg}) - Pr_{MCS}(y < 6 \text{deg})|}{Pr_{MCS}(y < 6 \text{deg})} \times 100\%.$$ 

Table 3: The uncertainty analysis results of the Fortini’s clutch example

| Method       | $N_{gd}$ | Mean   | Standard deviation | Skewness | Kurtosis | $Pr(y < 6\text{deg})$ | $\varepsilon_{Pr}$ |
|--------------|----------|--------|--------------------|----------|----------|----------------------|-------------------|
| MCS          | $10^6$   | 0.1219 | 0.0118             | -0.3156  | 3.2763   | 0.07881              | -                 |
| 5$^n$ UDR    | 17       | 0.1219 | 0.0117             | -0.1498  | 3.0669   | 0.07452              | 5.4435%           |
| 2-order PCE  | 81       | 0.1219 | 0.0119             | -0.3062  | 3.1324   | 0.08125              | 3.0961%           |
| 3-order PCE  | 256      | 0.1219 | 0.0118             | -0.3411  | 3.3112   | 0.07858              | 0.2918%           |
| 4-order PCE  | 625      | 0.1219 | 0.0118             | -0.3164  | 3.3361   | 0.07712              | 2.1444%           |
| Deep aPCE    | 17       | 0.1219 | 0.0118             | -0.2737  | 3.0950   | 0.07857              | 0.2969%           |
|              | 30       | 0.1219 | 0.0118             | -0.2829  | 3.1316   | 0.07872              | 0.1053%           |
|              | 40       | 0.1219 | 0.0118             | -0.3204  | 3.1809   | 0.07889              | 0.1015%           |

The units of mean and standard deviation are both ‘rad’.

According to Table 3, all methods can calculate the mean of contact angle $y$ correctly compared with the mean of the MCS method. When $N_{gd} = 17$, the standard deviation calculated by the Deep aPCE method is

23
0.118, consistent with the MCS method’s results. However, the standard deviation of 5\(^a\) UDR is 0.117 rather than 0.118. Besides, the skewness and kurtosis of the Deep aPCE method are closer to the MCS method’s results than 5\(^a\) UDR method. Apparently, the accuracy of the Deep aPCE method is more exact than 5\(^a\) UDR method when \(N_{gd} = 17\). When \(N_{gd} = 30\), the estimation error rate \(\varepsilon_{Pr}\) = 0.1053\% of the Deep aPCE method is already smaller than all other methods (The smallest \(\varepsilon_{Pr}\) value of other methods is 0.2918\% calculated by 3-order PCE method (256 data)). When \(N_{gd} = 40\), the estimation error rate \(\varepsilon_{Pr}\) = 0.1015\% of the Deep aPCE method is the smallest among all methods.

In order to quantitatively evaluate the accuracy of the Deep aPCE method, the determination coefficient \(R^2\) and the error \(e\) are calculated as shown in Table 4. Refer to Table 4, when \(N_{gd} = 17\), the determination coefficient \(R^2\) is 0.99961 and the error \(e\) is 0.00191. Thus, the Deep aPCE method only requires 17 labeled training data to estimate the PDF of contact angle \(y\) very accurately. With the amounts of labeled training data increasing, the determination coefficient \(R^2\) is improved gradually, and the error \(e\) is decreased gradually. When \(N_{gd} = 40\), the determination coefficient \(R^2\) and the error \(e\) are 0.99969 and 0.00171, respectively. Thus, the Deep aPCE method can estimate the contact angle \(y\) very accurately with little labeled training data.

Table 4: The determination coefficient \(R^2\) and the error \(e\) of the Deep aPCE method in the Fortini’s clutch example

| \(N_{gd}\) | 17  | 30  | 40  |
|--------|-----|-----|-----|
| \(R^2\) | 0.99961 | 0.99965 | 0.99969 |
| \(e\)   | 0.00191 | 0.00182 | 0.00171 |

Besides, based on kernel density estimation (KDE), the estimated probability density functions (PDFs) of contact angle \(y\) by the Deep aPCE method (17, 30, and 40 labeled training data) and MCS method are shown in Fig.11. The red curves (solid line) are the results of the Deep aPCE method, and the green curves (dash line) are the results of the MCS method. According to Fig.11, the PDF curves of contact angle \(y\) by the Deep aPCE method are very closer to the MCS method’s results for different amounts of labeled training data.

![Figure 11: The estimated probability density functions of contact angle \(y\) by the Deep aPCE method (17, 30, and 40 labeled training data) and MCS method for the Fortini’s clutch example.](image_url)

(a) 17 labeled training data  
(b) 30 labeled training data  
(c) 40 labeled training data
4.2. Example 2: Elastic cantilever beam

The second example considers an elastic cantilever beam \[6, 53\] subjected to two concentrated loads and one distributed load as shown in Fig.12. In this example, the displacement of point B is of interest, and its limit state function with respect to the displacement threshold $\Delta_{\text{lim}}$ is

$$G(q, F_1, F_2, E, I, L, \Delta_{\text{lim}}) = \Delta_{\text{lim}} - \left( \frac{qL^4}{8EI} + \frac{5F_1L^3}{48EI} + \frac{F_2L^3}{3EI} \right),$$

where the statistical properties of random variables $q$, $F_1$, $F_2$, $E$, $I$, $L$, and $\Delta_{\text{lim}}$ are shown in Table 5.

**Table 5: Statistical properties of random variables in the cantilever beam example**

| Variable | Description     | Mean       | C.O.V. | Distribution |
|----------|-----------------|------------|--------|--------------|
| $q$      | Distributed load| 50.0 N/mm  | 0.15   | Gumbel       |
| $F_1$    | Concentrated load| $7.0 \times 10^4$ N | 0.18 | Gumbel       |
| $F_2$    | Concentrated load| $1.0 \times 10^5$ N | 0.20 | Gumbel       |
| $E$      | Elastic modules | $2.6 \times 10^5$ MPa | 0.12 | Lognormal    |
| $I$      | Moment of inertia| $5.3594 \times 10^8$ mm$^4$ | 0.10 | Normal       |
| $L$      | Length of beam  | $3.0 \times 10^3$ mm | 0.05 | Normal       |
| $\Delta_{\text{lim}}$ | Threshold | 30.0 mm | 0.30 | Lognormal    |

C.O.V. = the coefficient of variation

![Figure 12: Elastic cantilever beam](image)

**Solving Deep aPCE model.** Like Example 1, the Deep aPCE model can be constructed for this elastic cantilever beam example. Besides, an MLP model with 7 inputs and 36 outputs is adopted to solve the expansion coefficients $c(\xi; \theta)$, and the neuron numbers of 5 hidden layers are 64, 128, 256, 256, and 256, respectively. In this example, $10^5$ non-labeled training data make up data set $D_{\text{ce}}$, i.e., $N_{\text{ce}} = 10^5$, and the number $N_{\text{gd}}$ of labeled training data is 40, 50, 60, 70, 80, and 90, respectively. Besides, the maximum training epoch $ep_{\text{max}}$ is set to be 7000, and the initial learning rate $\eta$ is set to be 0.01. During the model training process, the learning rate $\eta$ is scaled by 0.7 times every 300 epochs, i.e.,

$$\eta \leftarrow 0.7 \times \eta, \text{ if } (ep \mod 300) = 1.$$
Results analysis. Based on the trained Deep aPCE model \(D_{\text{APC}}(\xi; \theta)\), the uncertainty analysis results of this example are shown in Table 6. The results by MCS method (10^7 runs) are regarded as the truth results for comparison. Besides, to validate the effectiveness of Deep aPCE method, this example is also solved by 3-order CDA-DRM-SPCE method [6], 3-order LARS-PCE method [6], 3-order, 4-order and 5-order CDA-DRM-FPCE method [6], and 2-order OLS-PCE method [6], respectively.

Table 6: The uncertainty analysis results of the cantilever beam example

| Method               | \(N_{gd}\) | Mean (mm) | Standard deviation (mm) | Skewness | Kurtosis |
|----------------------|------------|-----------|-------------------------|----------|----------|
| MCS                  | 10^7       | 18.0946   | 9.5305                  | 0.7507   | 4.2713   |
| 3-order CDA-DRM-SPCE | 88         | 18.1026   | 9.4971                  | 0.7347   | 4.1541   |
| 3-order LARS-PCE     | 98         | 18.1093   | 9.4744                  | 0.7315   | 4.0851   |
| 3-order CDA-DRM-FPCE | 98         | 18.0909   | 9.5044                  | 0.7318   | 4.1519   |
| 4-order CDA-DRM-FPCE | 168        | 18.0865   | 9.5047                  | 0.7308   | 4.1533   |
| 5-order CDA-DRM-FPCE | 280        | 18.0865   | 9.5051                  | 0.7306   | 4.1533   |
| 2-order OLS-PCE      | 168        | 18.0845   | 9.4593                  | 0.6772   | 3.7715   |
| Deep aPCE            |            |           |                         |          |          |
| 40                   | 18.0546    | 9.5270    | 0.7345                  | 4.2370   |
| 50                   | 18.0799    | 9.5093    | 0.7570                  | 4.2706   |
| 60                   | 18.0642    | 9.5256    | 0.7505                  | 4.2633   |
| 70                   | 18.0965    | 9.5020    | 0.7614                  | 4.2837   |
| 80                   | 18.1046    | 9.5188    | 0.7589                  | 4.2712   |
| 90                   | 18.0914    | 9.5175    | 0.7478                  | 4.2511   |

Refer to Table 6, compared with the results of the MCS method, the relative errors of the Deep aPCE method are 0.22% on mean, 0.037% on standard deviation, 2.16% on skewness and 0.80% on kurtosis for \(N_{gd} = 40\), and the relative errors of 3-order CDA-DRM-SPCE method are 0.044% on mean, 0.35% on standard deviation, 2.13% on skewness and 2.74% on kurtosis for \(N_{gd} = 88\). Except that the relative error on mean and skewness of the Deep aPCE method is slightly larger than the 3-order CDA-DRM-SPCE method, the relative errors of the Deep aPCE method are far less than the 3-order CDA-DRM-SPCE method. What’s more, it is noteworthy that the amount of labeled training data used by the Deep aPCE method (40 data) are less than half that of the 3-order CDA-DRM-SPCE method (88 data). As the amount of labeled training data increases, the accuracy of the Deep aPCE method is improved gradually. Especially when \(N_{gd} = 70\), the Deep aPCE method’s relative errors are 0.010% on mean, 0.30% on standard deviation, 1.43% on skewness, and 0.29% on kurtosis. For 5-order CDA-DRM-FPCE method \((N_{gd} = 280)\), the relative errors are 0.045% on mean, 0.27% on standard deviation, 2.68% on skewness and 2.76% on kurtosis. The relative error on the standard deviation of the Deep aPCE method is only 0.03% larger than the 3-order CDA-DRM-SPCE method. However, the relative errors of the Deep aPCE method are much smaller than the 5-order CDA-DRM-FPCE method for mean, skewness, and kurtosis. Besides, the most important is that the amount of labeled training data is much smaller than the 5-order CDA-DRM-FPCE method.
data used by the Deep aPCE method is only a quarter of that of the 5-order CDA-DRM-FPCE method. Thus, compared with other methods in Table 6, the Deep aPCE method can approximate the limit state function very accurately with less labeled training data in this cantilever beam example.

To quantitatively evaluate the accuracy of the Deep aPCE method, the determination coefficient $R^2$ and the error $e$ are calculated as shown in Table 7. Refer to Table 7, when $N_{gd} = 40$, the determination coefficient $R^2$ is 0.99888 and the error $e$ is 0.01561. With the amounts of labeled training data increasing, the determination coefficient $R^2$ is improved gradually, and the error $e$ is decreased gradually. When $N_{gd} = 90$, the determination coefficient $R^2$ and the error $e$ are 0.99986 and 0.00545, respectively. Thus, the Deep aPCE method can estimate the limit state function value very accurately with little labeled training data.

Table 7: The determination coefficient $R^2$ and the error $e$ of the Deep aPCE method in the cantilever beam example

| $N_{gd}$ | 40  | 50  | 60  | 70  | 80  | 90  |
|--------|-----|-----|-----|-----|-----|-----|
| $R^2$  | 0.99888 | 0.99976 | 0.99971 | 0.99981 | 0.99986 | 0.99986 |
| $e$    | 0.01561 | 0.00729 | 0.00795 | 0.00641 | 0.00547 | 0.00545 |

Besides, based on KDE, the estimated PDFs of limit state function value by the Deep aPCE method (40, 50, 60, 70, 80, and 90 labeled training data) and the MCS method are shown in Fig.13. The red curves (solid line) are the results of the Deep aPCE method, and the green curves (dash line) are the results of the MCS method. According to Fig.13, the PDF curves of the limit state function value by the Deep aPCE method are very closer to the MCS method’s results for different amounts of labeled training data.

![PDF curves of the limit state function value](image)

Figure 13: The estimated probability density functions of the limit state function value by the Deep aPCE method (40, 50, 60, 70, 80, and 90 labeled training data) and the MCS method for the elastic cantilever beam example.
Effectiveness analysis of cost function. To validate the effectiveness of two absolute errors $L_{ce}^1 (\xi^J; \theta)$ and $L_{ce}^{2M} (\xi^J; \theta)$ in training the Deep aPCE model, this numerical example compares two Deep aPCE models $DAPC^J (\xi; \theta)$ and $DAPC^{L, \theta} (\xi; \theta)$, where $DAPC^J (\xi; \theta)$ is trained by the cost function $J (\theta)$ based on both the labeled training data and the non-labeled training data, and $DAPC^{L, \theta} (\xi; \theta)$ is trained by the mean absolute error $L_{gd} (\xi_{l}^{gd}, y_{l}^{gd}; \theta)$ based on the labeled training data only. Compared with the results of MCS method, the relative errors of two models’s uncertainty analysis results are shown in Table 8. Apparently, for $N_{gd} = 40, 50, 60, 70, 80, 90$, the relative errors on mean, standard deviation, skewness and kurtosis of the model $DAPC^J (\xi; \theta)$ are far less than the model $DAPC^{L, \theta} (\xi; \theta)$. Besides, the determination coefficients $R^2$ and the errors $e$ of the Deep aPCE models $DAPC^J (\xi; \theta)$ and $DAPC^{L, \theta} (\xi; \theta)$ are shown in Fig.14. For $N_{gd} = 40, 50, 60, 70, 80, 90$, the determination coefficients $R^2$ of the Deep aPCE model $DAPC^J (\xi; \theta)$ are closer to 1 than the model $DAPC^{L, \theta} (\xi; \theta)$, and the the errors $e$ of the Deep aPCE model $DAPC^J (\xi; \theta)$ are closer to 0 than the model $DAPC^{L, \theta} (\xi; \theta)$. Based on KDE, the estimated PDFs of limit state function value by the Deep aPCE models $DAPC^J (\xi; \theta)$ and $DAPC^{L, \theta} (\xi; \theta)$ (40, 50, and 60 labeled training data) are shown in Fig.15, where the green curves (dash line) are the results of the MCS method. According to Fig.15, the PDF curves of the limit state function value by model $DAPC^J (\xi; \theta)$ are closer to the MCS method’s results than model $DAPC^{L, \theta} (\xi; \theta)$.

Table 8: The uncertainty analysis results’ relative errors of the Deep aPCE models $DAPC^J (\xi; \theta)$ and $DAPC^{L, \theta} (\xi; \theta)$ in the cantilever beam example

| $N_{gd}$ | Model       | Mean       | Standard deviation | Skewness   | Kurtosis   |
|----------|-------------|------------|--------------------|------------|------------|
| 40       | $DAPC^{L, \theta} (\xi; \theta)$ | 26.65%     | 17.56%             | 263.33%    | 128.77%    |
|          | $DAPC^J (\xi; \theta)$          | 0.22%      | 0.037%             | 2.16%      | 0.80%      |
| 50       | $DAPC^{L, \theta} (\xi; \theta)$ | 16.99%     | 12.61%             | 182.31%    | 110.10%    |
|          | $DAPC^J (\xi; \theta)$          | 0.081%     | 0.22%              | 0.84%      | 0.016%     |
| 60       | $DAPC^{L, \theta} (\xi; \theta)$ | 14.40%     | 23.49%             | 138.87%    | 121.10%    |
|          | $DAPC^J (\xi; \theta)$          | 0.17%      | 0.051%             | 0.026%     | 0.19%      |
| 70       | $DAPC^{L, \theta} (\xi; \theta)$ | 6.08%      | 3.74%              | 23.93%     | 7.17%      |
|          | $DAPC^J (\xi; \theta)$          | 0.010%     | 0.30%              | 1.43%      | 0.29%      |
| 80       | $DAPC^{L, \theta} (\xi; \theta)$ | 7.86%      | 3.25%              | 31.30%     | 14.96%     |
|          | $DAPC^J (\xi; \theta)$          | 0.055%     | 0.12%              | 1.08%      | 0.0013%    |
| 90       | $DAPC^{L, \theta} (\xi; \theta)$ | 2.88%      | 1.51%              | 10.53%     | 10.48%     |
|          | $DAPC^J (\xi; \theta)$          | 0.018%     | 0.14%              | 0.39%      | 0.47%      |

In summary, according to the results of Table 8, Fig.14 and Fig.15, the accuracy of the Deep aPCE model $DAPC^J (\xi; \theta)$ trained by the cost function $J (\theta)$ are higher than the model $DAPC^{L, \theta} (\xi; \theta)$ trained by the mean absolute error $L_{gd} (\xi_{l}^{gd}, y_{l}^{gd}; \theta)$. According to Eq.(53), the cost function $J (\theta)$ is composed of
Figure 14: The determination coefficients $R^2$ and the errors $\varepsilon$ of the Deep aPCE models $D\text{APC}^J(\xi; \theta)$ and $D\text{APC}^{\text{ce}d}(\xi; \theta)$ in the cantilever beam example.

Figure 15: The estimated probability density functions of the limit state function value by the Deep aPCE models $D\text{APC}^J(\xi; \theta)$ and $D\text{APC}^{\text{ce}d}(\xi; \theta)$ (40, 50, and 60 labeled training data) and the MCS method for the elastic cantilever beam example. The blue lines in the first row’s three figures are the results obtained by $D\text{APC}^{\text{ce}d}(\xi; \theta)$, and the red lines in the second row’s three figures are the results obtained by $D\text{APC}^J(\xi; \theta)$.

where $L_{\text{gd}}^{\text{ce}d} \left( \xi^{\text{ce}d}; \theta \right)$, $L_{\text{ce}}^{\text{ce}d} \left( \xi^{\text{ce}d}; \theta \right)$, and $L_{\text{ce}}^{\text{ce}d} \left( \xi^{\text{ce}d}; \theta \right)$ are calculated by the non-labeled training data based on the properties of adaptive aPC. Therefore, only using a little amount of labeled training data, the applications of two absolute errors $L_{\text{ce}}^{\text{ce}d} \left( \xi^{\text{ce}d}; \theta \right)$ and $L_{\text{ce}}^{\text{ce}d} \left( \xi^{\text{ce}d}; \theta \right)$ in training the Deep aPCE model can obtain an accurate surrogate model.

4.3. Example 3: Three-bay six-storey planar frame

As shown in Fig.16, the third example considers a three-bay six-storey planar reinforced concrete frame structure with nonlinear constitutive laws. Same as the setting of reference [5, 6], the lateral top displacement $\Delta(X)$ of the three-bay six-storey planar reinforced concrete frame structure is calculated by the OpenSees software. The uni-axial material Concrete01 and Steel01 describe the nonlinear constitutive laws of concrete
and rebar, respectively, as shown in Fig.16. The limit state function with respect to the displacement threshold $\Delta_{\text{lim}}$ is

$$G(X) = \Delta_{\text{lim}} - \Delta(X),$$

(68)

where $\Delta_{\text{lim}} = 0.06m$, and $X$ are the random variables (including 24 independent random variables) of which the statistical properties are shown in Table 9.

Solving Deep aPCE model. Like Example 1, the Deep aPCE model can be constructed for this three-bay six-storey planar frame example. Besides, an MLP model with 24 inputs and 325 outputs is adopted to solve the expansion coefficients $c(\xi; \theta)$, and the neuron numbers of 5 hidden layers are 64, 128, 256, 512, and 512, respectively. In this example, $6 \times 10^5$ non-labeled training data make up the data set $D_{ce}$, i.e., $N_{ce} = 6 \times 10^5$, and the number $N_{gd}$ of labeled training data is 350, 370, 390 and 410, respectively. Besides, the maximum training epoch $ep_{max}$ is set to be 7000, and the initial learning rate $\eta$ is set to be 0.01. During the model training process, the learning rate $\eta$ is scaled by 0.7 times every 300 epochs, i.e.,

$$\eta \leftarrow 0.7 \times \eta, \quad \text{if } (ep \mod 300) = 1.$$

Results analysis. Based on the trained Deep aPCE model $\mathcal{DAPC}(\xi; \theta)$, the uncertainty analysis results of this example are shown in Table 10. The results by MCS method ($5 \times 10^5$ runs) are regarded as the truth results for comparison. Besides, to validate the effectiveness of Deep aPCE method, this example is also solved by 4-order and 5-order CDA-DRM-FPCE method [6], respectively.

Refer to Table 10, the Deep aPCE method can calculate the mean and the standard deviation of the limit state function value correctly for $N_{gd} = \{350, 370, 390, 410\}$ compared with the results of MCS. However, the
Table 9: Statistical properties of random variables in the three-bay six-storey planar frame

| Variable | Description                                      | Mean    | C.O.V. | Distribution |
|----------|--------------------------------------------------|---------|--------|--------------|
| $f_{cc}$ | Confined concrete compressive strength           | 35 MPa  | 0.10   | Normal       |
| $\varepsilon_{cc}$ | Confined concrete strain at maximum strength | 0.005   | 0.05   | Normal       |
| $f_{cu}$ | Confined concrete crushing strength              | 25 MPa  | 0.10   | Normal       |
| $\varepsilon_{cu}$ | Confined concrete strain at crushing strength | 0.02    | 0.05   | Normal       |
| $f_{c}$  | Unconfined concrete compressive strength         | 27 MPa  | 0.10   | Normal       |
| $f_{u}$  | Unconfined concrete crushing strength            | 0 MPa   | -      | Deterministic|
| $\varepsilon_{c}$ | Unconfined concrete strain at maximum strength | 0.002   | 0.05   | Normal       |
| $\varepsilon_{u}$ | Unconfined concrete strain at crushing strength | 0.006   | 0.05   | Normal       |
| $f_{y}$  | Yield strength of rebar                          | 400 MPa | 0.10   | Normal       |
| $E_0$    | Initial elastic modulus of rebar                 | 200 GPa | 0.10   | Normal       |
| $b$      | Strain-hardening ratio of rebar                  | 0.007   | 0.05   | Normal       |
| $q_1$    | Uniform load                                     | 21.41 kN/m | 0.25 | Lognormal   |
| $q_2$    | Uniform load                                     | 11.48 kN/m | 0.25 | Lognormal   |
| $q_3$    | Uniform load                                     | 22.68 kN/m | 0.25 | Lognormal   |
| $q_4$    | Uniform load                                     | 12.18 kN/m | 0.25 | Lognormal   |
| $Q_1$    | External force                                   | 42.43 kN | 0.25   | Lognormal   |
| $Q_2$    | External force                                   | 68.25 kN | 0.25   | Lognormal   |
| $Q_3$    | External force                                   | 44.03 kN | 0.25   | Lognormal   |
| $Q_4$    | External force                                   | 71.35 kN | 0.25   | Lognormal   |
| $F_1$    | External force                                   | 10 kN   | 0.25   | Lognormal   |
| $F_2$    | External force                                   | 20 kN   | 0.25   | Lognormal   |
| $F_3$    | External force                                   | 30 kN   | 0.25   | Lognormal   |
| $F_4$    | External force                                   | 40 kN   | 0.25   | Lognormal   |
| $F_5$    | External force                                   | 50 kN   | 0.25   | Lognormal   |
| $F_6$    | External force                                   | 60 kN   | 0.25   | Lognormal   |

standard deviations of both the 4-order CDA-DRM-FPCE method and the 5-order CDA-DRM-FPCE method are 0.0067 rather than 0.0068. Compared with the MCS method’s results, the relative errors of the Deep aPCE method are 1.94% on skewness and 0.80% on kurtosis for $N_{gd} = 350$, the relative errors of 4-order CDA-DRM-FPCE method are 8.28% on skewness and 3.17% on kurtosis for $N_{gd} = 378$, and the relative errors of 5-order CDA-DRM-FPCE are 7.41% on skewness and 1.89% on kurtosis for $N_{gd} = 630$. Apparently, the Deep aPCE method uses only 350 labeled training data to obtain higher accuracy than the 4-order CDA-DRM-FPCE method (378 labeled training data) and the 5-order CDA-DRM-FPCE method (630 labeled training data). What’s more, with the amounts of the labeled training data increasing, the accuracy of the Deep aPCE
Table 10: The uncertainty analysis results of the three-bay six-storey planar reinforced concrete frame example

| Method                  | \(N_{gd}\) | Mean   | Standard deviation | Skewness | Kurtosis |
|-------------------------|-------------|--------|--------------------|----------|----------|
| MCS                     | \(5 \times 10^5\) | 0.0209 | 0.0068             | −0.6705  | 3.8955   |
| 4-order CDA-DRM-FPCE    | 378         | 0.0209 | 0.0067             | −0.6150  | 3.7722   |
| 5-order CDA-DRM-FPCE    | 630         | 0.0209 | 0.0067             | −0.6208  | 3.8220   |
| Deep aPCE               |             |        |                    |          |          |
|                         | 350         | 0.0209 | 0.0068             | −0.6835  | 3.8644   |
|                         | 370         | 0.0209 | 0.0068             | −0.6882  | 3.9078   |
|                         | 390         | 0.0209 | 0.0068             | −0.6873  | 3.9049   |
|                         | 410         | 0.0209 | 0.0068             | \(0.6752\) | 3.8927   |

The units of mean and standard deviation are both ‘m’.

method is improved gradually. For \(N_{gd} = 410\), the relative errors of the Deep aPCE method are only 0.70% on skewness and 0.07% on kurtosis. Thus, the Deep aPCE method can approximate the limit state function accurately with little labeled training data.

To quantitatively evaluate the accuracy of the Deep aPCE method, the determination coefficient \(R^2\) and the error \(e\) are calculated as shown in Table 11. Refer to Table 11, when \(N_{gd} = 350\), the determination coefficient \(R^2\) is 0.99533 and the error \(e\) is 0.02124. With the amounts of the labeled training data increasing, the determination coefficient \(R^2\) is improved gradually, and the error \(e\) is decreased gradually. When \(N_{gd} = 650\), the determination coefficient \(R^2\) and the error \(e\) are 0.99965 and 0.00585, respectively. Thus, the Deep aPCE method can estimate the limit state function value very accurately with little labeled training data.

Table 11: The determination coefficient \(R^2\) and the error \(e\) of the Deep aPCE method in the three-bay six-storey planar frame example

| \(N_{gd}\) | 350  | 370  | 410  | 450  | 490  | 530  | 570  | 600  | 650  |
|------------|------|------|------|------|------|------|------|------|------|
| \(R^2\)    | 0.99533 | 0.99717 | 0.99927 | 0.99932 | 0.99949 | 0.99950 | 0.99957 | 0.99959 | 0.99965 |
| \(e\)      | 0.02124 | 0.01654 | 0.00837 | 0.00811 | 0.00701 | 0.00696 | 0.00647 | 0.00629 | 0.00585 |

Besides, based on KDE, the estimated PDFs of limit state function value by the Deep aPCE method (350, 370, 410, 450, 490, 530, 570, 600, and 650 labeled training data) and MCS method are shown in Fig.17. The red curves (solid line) are the results of the Deep aPCE method, and the green curves (dash line) are the results of the MCS method. According to Fig.17, the PDF curves of the limit state function value by the Deep aPCE method are very closer to the MCS method’s results for different amounts of labeled training data.
4.4. Example 4: High-dimensional example

The fourth example investigates the uncertainty quantification of a high-dimensional problem which was proposed by Rackwitz [54]. The analytical limit state function is

\[
G(x_1, x_2, \ldots, x_n) = n + 3\sigma\sqrt{n} - \sum_{i=1}^{n} x_i,
\]

where the random variables \(x_i\) \((i = 1, 2, \ldots, n)\) are independent, identically and lognormally distributed. In this paper, the number of random variables is 40, i.e., \(n = 40\), and the means and standard deviations are \(\mu_{x_i} = 1\) and \(\sigma_{x_i} = 0.2\) \((i = 1, 2, \ldots, n)\), respectively.

**Solving Deep aPCE model.** Like Example 1, the Deep aPCE model can be constructed for this high-dimensional example. Besides, an MLP model with 40 inputs and 861 outputs is adopted to solve the expansion coefficients \(c(\xi; \theta)\), and the neuron numbers of 5 hidden layers are 100, 600, 800, 900, and 900, respectively. In this example, \(5 \times 10^4\) non-labeled training data make up the data set \(D_{ce}\), i.e., \(N_{ce} = 5 \times 10^4\), and the
number \( N_{gd} \) of labeled training data is 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, and 2000, respectively. Besides, the maximum training epoch \( ep_{\text{max}} \) is set to be 2000, and the initial learning rate \( \eta \) is set to be 0.01. During the model training process, the learning rate \( \eta \) is scaled by 0.6 times every 200 epochs, i.e.

\[
\eta \leftarrow 0.6 \times \eta, \quad \text{if } (ep \mod 200) = 1.
\]

**Results analysis.** Based on the trained Deep aPCE model \( \mathcal{DAPC}(\xi; \theta) \), the uncertainty analysis results of this example are shown in Table 12. In this example, if the value of the limit state function is less than 0, the failure will happen. Thereby, the failure probability \( P_f \) is estimated by the Deep aPCE method as shown in Table 13. In Table 12, the MCS method’s results (10^6 runs) are regarded as the truth results for comparison. Besides, to validate the effectiveness of the Deep aPCE method, this example is also solved by the SIR-PCE method [28]. Compared with the MCS method’s results, \( \epsilon_{P_f} \) denote the estimation error rate of failure probability \( P_f \), i.e.,

\[
\epsilon_{P_f} = \frac{|P_f - P_f^{MC}\text{S}|}{P_f^{MC}\text{S}} \times 100\%.
\]

### Table 12: The uncertainty analysis results of the high-dimensional example

| Method   | \( N_{gd} \) | Mean    | Standard deviation | Skewness | Kurtosis | \( \epsilon_{P_f} \) | \( R^2 \) |
|----------|--------------|---------|--------------------|----------|----------|-----------------------|----------|
| MCS      | \( 10^6 \)   | 3.7944  | 1.2634             | -0.0964  | 3.0077   | -                     | -        |
| SIR-PCE  | 1200         | 3.7962  | 1.2527             | -0.1115  | 3.0495   | 8.20%                 | 0.981000 |
|          | 1600         | 3.7947  | 1.2566             | -0.0925  | 3.0121   | 0.50%                 | 0.980500 |
|          | 2000         | 3.7948  | 1.2626             | -0.0915  | 3.0234   | 0.50%                 | 0.981700 |
| **Deep aPCE** | 1200         | 3.7854  | 1.2634             | -0.0973  | 3.0089   | 2.30%                 | 0.999611 |
|          | 1300         | 3.7943  | 1.2635             | -0.0967  | 3.0079   | 0.16%                 | 0.999984 |
|          | 1400         | 3.7938  | 1.2632             | -0.0962  | 3.0062   | **0.05%**             | 0.999986 |
|          | 1500         | 3.7939  | 1.2632             | -0.0968  | 3.0075   | 0.27%                 | 0.999993 |
|          | 1600         | 3.7940  | 1.2633             | -0.0962  | 3.0073   | **0.05%**             | 0.999992 |
|          | 1700         | 3.7944  | 1.2637             | -0.0962  | 3.0065   | 0.43%                 | 0.999993 |
|          | 1800         | 3.7942  | 1.2631             | -0.0964  | 3.0074   | 0.11%                 | 0.999995 |
|          | 1900         | 3.7943  | 1.2633             | -0.0963  | 3.0080   | **0.00**              | **0.999996** |
|          | 2000         | 3.7943  | 1.2633             | -0.0963  | 3.0074   | **0.00**              | **0.999996** |

For \( N_{gd} = 1200 \), the relative errors of the Deep aPCE method are 0.24% on mean, no error on standard deviation, 0.93% on skewness and 0.04% on kurtosis, and the relative errors of the SIR-PCE method are 0.05% on mean, 0.01% on standard deviation, 15.67% on skewness and 1.39% on kurtosis. Except that the relative error on the mean of the Deep aPCE method is slightly larger than the SIR-PCE method, the relative errors of the Deep aPCE method are less than the SIR-PCE method. Especially, the relative error on the skewness of the Deep aPCE method (0.93%) is much smaller than the SIR-PCE method (15.67%). Besides, the estimation error rate \( \epsilon_{P_f} \) of the Deep aPCE method is 2.30% less than 8.20% of the SIR-PCE method. As shown in
Table 12, compared with the MCS method’s results, the estimation accuracy of mean, standard deviation, skewness, and kurtosis increases with the number $N_{gd}$ of labeled training data increasing for both Deep aPCE method and SIR-PCE method. However, the estimation accuracy of the Deep aPCE method is higher than the SIR-PCE method. Apparently, the accuracy of the Deep aPCE method based on 1300 labeled training data is higher than the SIR-PCE method based on 2000 labeled training data. For $N_{gd} = 1900$, the Deep aPCE method has already no estimation error for the failure probability $P_f$ as shown in Table 12 and Table 13. However, the SIR-PCE method still has an 8.20% estimation error for the failure probability $P_f$ based on 2000 labeled training data.

Table 13: Estimation of the failure probability in the high-dimensional example

| $N_{gd}$  | 10$^6$ | 1200 | 1300 | 1400 | 1500 | 1600 | 1700 | 1800 | 1900 | 2000 |
|-----------|-------|------|------|------|------|------|------|------|------|------|
| **Deep aPCE ($\times 10^{-3}$)** | - | 1.899 | 1.859 | 1.857 | 1.861 | 1.855 | 1.848 | 1.854 | **1.856** | 1.856 |
| **MCS ($\times 10^{-3}$)** | 1.856 |

When $N_{gd} = 1200$, the determination coefficient $R^2$ of the Deep aPCE method has up to 0.999611, but the determination coefficient $R^2$ of the SIR-PCE method is only 0.981000. Besides, with the amounts of labeled training data increasing, the determination coefficient $R^2$ of the Deep aPCE method is improved gradually. When $N_{gd} = 2000$, as shown in Table 12, the determination coefficient $R^2$ of the Deep aPCE method is 0.999996 which is more than 0.981700 calculated by the SIR-PCE method. Apparently, the Deep aPCE method can estimate the limit state function value more accurately than the SIR-PCE method.

To show the accuracy of the Deep aPCE method more intuitively, based on KDE, the estimated PDFs of limit state function value by the Deep aPCE method (1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, and 2000 labeled training data) and the MCS method are shown in Fig.18. The red curves (solid line) are the results of the Deep aPCE method, and the green curves (dash line) are the results of the MCS method. According to Fig.18, the PDF curves of limit state function value by the Deep aPCE method are very closer to the MCS method’s results for different amounts of labeled training data.

In summary, with the same amount of labeled training data, the accuracy of the Deep aPCE method is higher compared with the SIR-PCE method. Besides, the Deep aPCE method can estimate the limit state function value of this high-dimensional example very accurately with little labeled training data.

5. Engineering application

5.1. Engineering background

In this section, a practical engineering problem, i.e., the first-order frequency uncertainty analysis of the micro-satellite TianTuo-3 (TT-3) frame structure (Fig.19), is used to validate the effectiveness of the proposed Deep aPCE method. As shown in Fig.19, the TT-3 satellite frame structure includes three parts, i.e., the TT-3 satellite, four support rods and one separating device.
To avoid resonance during launch, the TT-3 satellite frame structure’s first-order frequency must be more significant than 81.0 Hz. For TT-3’s frame structure, there are mainly six random variables (Table 14) that affect the first-order frequency calculation. Due to the existence of uncertainty factors, as shown in Table 14, the engineers need to quantify the first-order frequency uncertainty to guide the design of the TT-3’s frame structure. Generally, a high-fidelity FEA model is used to calculate the first-order frequency. However, to quantify the first-order frequency’s uncertainty, the FEA model has to be called many times, leading to high computational costs. Therefore, the proposed Deep aPCE model is adopted to quantify the first-order frequency’s uncertainty of TT-3’s frame structure to improve the computational efficiency.

5.2. Constructing surrogate model

Preparing data. According to engineering experience, each random variable is described by the normal distribution, and the corresponding mean and standard deviation are determined by ‘3σ’ rule based on the variation range of the random variable. For example, the mean \( \mu_{p_1} \) and the standard deviation \( \sigma_{p_1} \) of aluminum...
Table 14: Uncertainty parameters of the TT-3 satellite frame structure

| Variable | Description                  | True value | Uncertainty source | V.R.   |
|----------|------------------------------|------------|--------------------|--------|
| $\rho_1$ | Aluminum alloy density       | 2.69 g/cm$^3$ | M.M.C.             | ±1%    |
| $\rho_2$ | Spring steel density         | 7.85 g/cm$^3$ | M.M.C.             | ±1%    |
| $\rho_3$ | Titanium alloy density       | 4.43 g/cm$^3$ | M.M.C.             | ±1%    |
| $E_1$    | Aluminum elastic modulus     | 68.9 GPa   | E.T.C.             | ±10%   |
| $E_2$    | Spring steel elastic modulus | 200 GPa    | E.T.C.             | ±3%    |
| $E_3$    | Titanium alloy elastic modulus | 113.8 GPa | E.T.C.             | ±5%    |

V.R. = the variation range; M.M.C. = the material molding conditions; E.T.C. = the environmental temperature change.

alloy density $\rho_1$ are calculated by

$$\begin{cases}
\mu_{\rho_1} - 3\sigma_{\rho_1} = 2.69 \times (1 - 0.01) \\
\mu_{\rho_1} + 3\sigma_{\rho_1} = 2.69 \times (1 + 0.01)
\end{cases} \Rightarrow \begin{cases}
\mu_{\rho_1} = 2.69 \\
\sigma_{\rho_1} = 8.97 \times 10^{-3}.
\end{cases} \tag{70}$$

Similar to Eq.(70), the means and the standard deviation of the rest of the random variables can be calculated, and the results are shown in Table 15.

Based on the parameters in Table 15, the value of six random variables are sampled by Latin Hypercube Sampling. Then, based on the ABAQUS software, the corresponding first-order frequency is calculated by the FEA model of TT-3’s frame structure, as shown in Fig.20. There are two kinds of training data sets and a test data set, i.e., the labeled training data set $D_{gd} = \{ (X^{gd}_l, y^{gd}_l) | l = 1, 2, \cdots, N_{gd} \}$, the non-labeled training data set $D_{ce} = \{ X^{ce}_{l'} | l' = 1, 2, \cdots, N_{ce} \}$ and the test data set $D_{test} = \{ (X^{test}_l, y^{test}_l) | l = 1, 2, \cdots, N_{test} \}$, where $X = \{ \rho_1, \rho_2, \rho_3, E_1, E_2, E_3 \}$. The inputs $X^{gd}_l$, $X^{ce}_{l'}$ and $X^{test}_l$ are normalized firstly. Thereby, three data sets are transformed to be $D_{gd} = \{ (\xi^{gd}_l, \eta^{gd}_l) | l = 1, 2, \cdots, N_{gd} \}$, $D_{ce} = \{ \xi^{ce}_{l'} | l' = 1, 2, \cdots, N_{ce} \}$, $D_{test} = \{ (\xi^{test}_l, \eta^{test}_l) | l = 1, 2, \cdots, N_{test} \}$, respectively. In this example, $10^5$ non-labeled training data make up the
Table 15: Statistical properties of random variables in the TT-3 satellite frame structure

| Variable | mean       | Standard deviation | Distribution |
|----------|------------|--------------------|--------------|
| $\rho_1$ | 2.69 g/cm$^3$ | $8.97 \times 10^{-3}$ g/cm$^3$ | Normal       |
| $\rho_2$ | 7.85 g/cm$^3$ | $2.617 \times 10^{-2}$ g/cm$^3$ | Normal       |
| $\rho_3$ | 4.43 g/cm$^3$ | $1.477 \times 10^{-2}$ g/cm$^3$ | Normal       |
| $E_1$   | $6.89 \times 10^4$ MPa | $2.29667 \times 10^3$ MPa | Normal       |
| $E_2$   | $2.00 \times 10^5$ MPa | $2.00 \times 10^3$ MPa | Normal       |
| $E_3$   | $1.138 \times 10^5$ MPa | $1.89667 \times 10^3$ MPa | Normal       |

...data set $D_{ce}$, i.e., $N_{ce} = 10^5$. Besides, the number $N_{gd}$ of labeled training data is 30, 40, 50, 60, and 70, respectively. Due to the computationally expensive FEA model (about 5 minutes for one single FEA simulation), 600 data are used to be the test data in this paper.

Figure 20: The FEA model of the TT-3 satellite frame structure

Deep aPCE model of TT-3 satellite frame structure. The random input variable $X = \{\rho_1, \rho_2, \rho_3, E_1, E_2, E_3\}$ is normalized to be $\xi = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6\}$. Thus, a 2-order Deep aPCE model $\mathcal{DAPC}(\xi; \theta)$ is used to be the surrogate model of the first-order frequency of TT-3’s frame structure, i.e.,

$$\hat{y}^{(2)}(\xi) = \sum_{i=1}^{28} c_i(\xi; \theta) \Phi_i(\xi),$$

where $c_i(\xi; \theta)$ is calculated by the MLP model $\mathcal{NN}(\xi; \theta)$. In this example, an MLP model with seven layers is adopted, as shown in Fig.21, and this MLP model has 6 inputs and 28 outputs. Besides, the neuron numbers of 5 hidden layers are 64, 128, 128, 128 and 128, respectively.

Deep aPCE model training. Based on the labeled training data set $D_{gd} = \{\left(\xi_{gd}^l, y_{gd}^l\right) | l = 1, 2, \cdots, N_{gd}\}$ ($N_{gd} = 30, 40, 50, 60, 70$) and the non-labeled training data set $D_{ce} = \{\xi_{ce}^{l'} | l' = 1, 2, \cdots, 10^5\}$, the Deep aPCE...
5.3. Results analysis and discussion

Based on the trained Deep aPCE model $D_{\mathcal{A}PCE}(\xi; \theta)$, the uncertainty analysis results are shown in Table 16. When $N_{gd} = 30$, the relative errors of Deep aPCE method are no error on mean, 1.32% on standard deviation, 7.71% on skewness, and 0.038% on kurtosis. Besides, the error $e$ and the determination coefficient $R^2$ are 0.00019 and 0.99881, respectively. Thus, based on 30 labeled training data, the Deep aPCE model can fit the first-order frequency model of TT-3’s frame structure with only a bit of error. As shown in Table 16, compared with the FEA-MCS method’s results, the estimation accuracies of the Deep aPCE method’s mean, standard deviation, skewness, and kurtosis increase with the number $N_{gd}$ of labeled training data. When $N_{gd} = 50$, the Deep aPCE method can accurately calculate the mean, standard deviation, and kurtosis of the first-order frequency. Especially, based on 70 labeled training data, the error $e$ is only 0.00003, and the determination coefficient $R^2$ has arrived at 0.99998. For five Deep aPCE models ($N_{gd} = 30, 40, 50, 60, 70$), the predictive absolute errors for 600 random inputs are shown in Fig.22, and their average absolute errors are only 0.01017, 0.00703, 0.00629 and 0.00165 as shown in Table 16, respectively. Refer to Fig.22, the predictive absolute error decreases gradually with the number $N_{gd}$ of labeled training data increasing.

Besides, the $k$-fold cross-validation technique is also used to validate that the Deep aPCE model can fit the first-order frequency model of TT-3’s frame structure perfectly. For $N_{gd} = 40, 50, 60, 70$, each labeled training data set is randomly divided into 5 subsets equally. Then, the Deep aPCE model is trained by four subsets and the remaining one subset is used to test the trained Deep aPCE model, which is repeated five times. The average absolute errors of 5 models for four kinds of labeled training data sets are shown in Table 17. In Table 17, ’$32+8$’ denotes that 32 data and 8 data are respectively used to train and test the Deep aPCE model in each training, ’$40+10$’, ’$48+12$’ and ’$56+14$’ denote similar meaning. The final average absolute errors are only 0.00432, 0.00153, 0.00082, and 0.00072, respectively. In summary, the Deep aPCE model can fit the first-order frequency model of TT-3’s frame structure well according to the results of Table 16, Table 17, and Fig.22.

According to the above analysis, the Deep aPCE model based on 70 labeled training data can accurately
Table 16: The uncertainty analysis results and the average absolute errors of the first-order frequency of TT-3’s frame structure

| Method   | $N_{gd}$ | Mean  | Standard deviation | Skewness | Kurtosis | $e$       | $R^2$   | A.A.E.       |
|----------|----------|-------|--------------------|----------|----------|----------|--------|--------------|
| FEA-MCS  | 600      | 82.84 | 0.453              | -0.0726  | 2.652    | -        | -      | -            |
| Deep aPCE| 30       | 82.84 | 0.447              | -0.0670  | 2.651    | 0.00019  | 0.99881 | 0.01017      |
|          | 40       | 82.84 | 0.447              | -0.0720  | 2.650    | 0.00011  | 0.99958 | 0.00703      |
|          | 50       | 82.84 | 0.453              | -0.0712  | 2.652    | 0.00015  | 0.99926 | 0.00629      |
|          | 60       | 82.84 | 0.453              | -0.0745  | 2.653    | 0.00005  | 0.99992 | 0.00352      |
|          | 70       | 82.84 | 0.453              | -0.0713  | 2.651    | 0.00003  | 0.99998 | 0.00165      |

A.A.E. = average absolute error. The units of mean and standard deviation are both 'Hz'.

Figure 22: The absolute error between the Deep aPCE method ($N_{gd} = 30, 40, 50, 60, 70$) and the FEA-MCS method in the TT-3 satellite frame structure example. The absolute error values are sorted in ascending order.

Table 17: The average absolute errors of 5-fold cross-validation in the TT-3’s frame structure example

| $N_{gd}$ | Model-1 | Model-2 | Model-3 | Model-4 | Model-5 | Average value |
|----------|---------|---------|---------|---------|---------|---------------|
| 40 (32+8) | 0.00071 | 0.00107 | 0.00105 | 0.01783 | 0.00096 | **0.00432**   |
| 50 (40+10) | 0.00116 | 0.00371 | 0.00101 | 0.00104 | 0.00073 | **0.00153**   |
| 60 (48+12) | 0.00064 | 0.00103 | 0.00092 | 0.00080 | 0.00071 | **0.00082**   |
| 70 (56+14) | 0.00068 | 0.00054 | 0.00087 | 0.00063 | 0.00088 | **0.00072**   |

estimate the TT-3 satellite frame structure’s first-order frequency. Thus, the MCS is performed $10^7$ times on the Deep aPCE model ($N_{gd} = 70$). Then, based on the $10^7$ results, the probability density function of the first-order frequency is estimated by the KDE method, as shown in Fig.23. According to section 5.1, the TT-3 satellite frame structure will be damaged due to resonance if the first-order frequency is less than 81.0 Hz during launch. Thus, the TT-3 satellite frame structure’s failure probability is 0.0000151. Apparently, the
failure probability is sufficiently small. Therefore, the design of the TT-3 satellite frame structure is reliable, which is consistent with the successful launch of micro-satellite TT-3.

![Probability Density Function](image)

Figure 23: The estimated probability density function of the first-order frequency of TT-3’s frame structure based on the Deep aPCE model ($N_{gd} = 70$)

6. Conclusions

For the uncertainty quantification of stochastic systems, the Deep aPCE method is proposed by making full use of the advantages of aPC and DL. On the one hand, by the adaptive expansion coefficients, which are fine-tuned dynamically by the MLP model for different random input variables, the Deep aPCE method can construct an accurate surrogate model by a lower order PCE. On the other hand, this paper makes full use of the non-labeled data to propose a cost function based on the adaptive arbitrary polynomial chaos expansion’s properties, based on which the Deep aPCE method only needs a little amount of labeled data to train the MLP model. To verify the proposed Deep aPCE method’s effectiveness, four numerical examples, including low, medium, and appropriate high dimensional stochastic systems, are used in this paper. Using less amount of labeled training data, the accuracy of the Deep aPCE method can be higher than that of the existed PCE methods. Thus, the Deep aPCE method can significantly reduce the calling times of computationally expensive model and is suitable for low, medium, and appropriate high dimensional stochastic systems. Besides, the Deep aPCE method is applied to the uncertainty analysis of an actual engineering problem, and the results show that the micro-satellite TT-3 frame structure’s design is reliable, which is consistent with the successful launch of micro-satellite TT-3.

In summary, the proposed Deep aPCE method has the following three advantages: (1) By adaptive expansion coefficients, the Deep aPCE method needs fewer labeled training data than the existed PCE methods to construct an accurate surrogate model; (2) Based on the properties of adaptive arbitrary polynomial chaos expansion, the Deep aPCE method can overcome the shortcoming that a large amount of labeled training data needed to train the MLP model; (3) Due to the application of DL, the proposed Deep aPCE method can
construct accurate surrogate models for the appropriate high dimensional stochastic systems. For the super-high dimensional stochastic systems, both the Deep aPCE method and the existed PCE methods may require more labeled training data for constructing surrogate models. Therefore, in future research, the authors will explore how to reduce the number of labeled training data in constructing the Deep aPCE model of super-high dimensional stochastic systems.

Acknowledgments

This work was supported by the Postgraduate Scientific Research Innovation Project of Hunan Province (No.CX20200006) and the National Natural Science Foundation of China (Nos.11725211 and 52005505).

Appendix A.

By the orthonormality of the multi-dimensional orthogonal basis \{\Phi_1(\xi), \Phi_2(\xi), \cdots, \Phi_M(\xi)\}, the following four equations can be obtained, i.e.,

\[
\int_{\xi \in \Omega^d} \Phi_1(\xi) \, d\Gamma(\xi) = 1 \quad (A.1)
\]

\[
\int_{\xi \in \Omega^d} \Phi_i(\xi) \, d\Gamma(\xi) = 0 \quad (i = 2, 3, \cdots, M.) \quad (A.2)
\]

\[
\int_{\xi \in \Omega^d} \Phi_i^2(\xi) \, d\Gamma(\xi) = 1 \quad (i = 1, 2, \cdots, M.) \quad (A.3)
\]

\[
\int_{\xi \in \Omega^d} \Phi_i(\xi) \Phi_j(\xi) \, d\Gamma(\xi) = 0 \quad (i \neq j; \ i = 1, 2, \cdots, M; \ j = 1, 2, \cdots, M.) \quad (A.4)
\]

According to Eq.(24), the mean of \(\hat{y}^{(p)}(\xi)\) is

\[
E \left[ \hat{y}^{(p)}(\xi) \right] = E \left\{ \sum_{i=1}^{M} \mathcal{C}_i(\xi; \theta) \Phi_i(\xi) \right\} \\
= \sum_{i=1}^{M} E \left[ \mathcal{C}_i(\xi; \theta) \Phi_i(\xi) \right] \quad (A.5)
\]

\[
= E \left[ \mathcal{C}_1(\xi; \theta) \Phi_1(\xi) \right] + \sum_{i=2}^{M} E \left[ \mathcal{C}_i(\xi; \theta) \Phi_i(\xi) \right].
\]
Refer to Eq.(26), $C_i (\xi; \theta) = E [C_i (\xi; \theta)] + \delta^E_i$, where $\delta^E_i$ is an infinitesimal for $\theta \to \theta^*$. Thus, Eq.(A.5) can be

$$E \left[ \hat{y}^{(p)} (\xi) \right] = E \left[ C_1 (\xi; \theta) \Phi_1 (\xi) \right] + \sum_{i=2}^{M} E \left[ C_i (\xi; \theta) \Phi_i (\xi) \right]$$

$$= E \left\{ E \left[ C_1 (\xi; \theta) \right] + \delta^E_1 \right\} \Phi_1 (\xi) + \sum_{i=2}^{M} E \left\{ E \left[ C_i (\xi; \theta) \right] + \delta^E_i \right\} \Phi_i (\xi)$$

$$= E \left\{ E \left[ C_1 (\xi; \theta) \right] \Phi_1 (\xi) \right\} + E \left\{ \delta^E_1 \Phi_1 (\xi) \right\} + \sum_{i=2}^{M} E \left\{ E \left[ C_i (\xi; \theta) \right] \Phi_i (\xi) \right\} + \sum_{i=2}^{M} E \left\{ \delta^E_i \Phi_i (\xi) \right\}$$

$$= \int_{\xi \in \Omega^d} E \left[ C_1 (\xi; \theta) \right] \Phi_1 (\xi) d\Gamma (\xi) + \sum_{i=2}^{M} \left[ \int_{\xi \in \Omega^d} E \left[ C_i (\xi; \theta) \right] \Phi_i (\xi) d\Gamma (\xi) \right]$$

$$+ \sum_{i=2}^{M} E \left[ \delta^E_i \Phi_i (\xi) \right]$$

$$= E \left[ C_1 (\xi; \theta) \right] \int_{\xi \in \Omega^d} \Phi_1 (\xi) d\Gamma (\xi) + \sum_{i=2}^{M} E \left[ C_i (\xi; \theta) \right] \int_{\xi \in \Omega^d} \Phi_i (\xi) d\Gamma (\xi)$$

$$+ \sum_{i=1}^{M} \left[ \frac{1}{N} \sum_{l=1}^{N} \delta^E_i \Phi_i (\xi_l) \right],$$

where $N$ is a finite positive integer. Due to $\delta^E_i (l = 1, 2, \cdots, N)$ are infinitesimals, $\delta^E_i \Phi_i (\xi_l)$ is an infinitesimal for $\theta \to \theta^*$. Therefore,

$$\delta^E = \sum_{i=1}^{M} \left[ \frac{1}{N} \sum_{l=1}^{N} \delta^E_i \Phi_i (\xi_l) \right]$$

is an infinitesimal for $\theta \to \theta^*$. On the basis of Eqs.(A.1), (A.2) and (A.7), Eq.(A.6) can be

$$E \left[ \hat{y}^{(p)} (\xi) \right] = E \left[ C_1 (\xi; \theta) \right] \int_{\xi \in \Omega^d} \Phi_1 (\xi) d\Gamma (\xi) + \sum_{i=2}^{M} E \left[ C_i (\xi; \theta) \right] \int_{\xi \in \Omega^d} \Phi_i (\xi) d\Gamma (\xi)$$

$$+ \sum_{i=1}^{M} \left[ \frac{1}{N} \sum_{l=1}^{N} \delta^E_i \Phi_i (\xi_l) \right]$$

$$= E \left[ C_1 (\xi; \theta) \right] + \delta^E.$$

Thus, the mean of $\hat{y}^{(p)} (\xi)$ approaches the mean of the first adaptive expansion coefficient $C_1 (\xi; \theta)$ for $\theta \to \theta^*$, i.e.,

$$E \left[ \hat{y}^{(p)} (\xi) \right] - E \left[ C_1 (\xi; \theta) \right] < \varepsilon_3, \quad (\forall \varepsilon_3 > 0).$$
According to Eq. (24), the variance of \( \hat{y}^{(p)} (\xi) \) is

\[
\text{Var} \left[ \hat{y}^{(p)} (\xi) \right] = E \left\{ \left[ \sum_{i=1}^{M} C_i (\xi; \theta) \Phi_i (\xi) \right]^2 \right\} - E^2 \left[ \hat{y}^{(p)} (\xi) \right]
\]

\[
= E \left\{ \sum_{i=1}^{M} C_i^2 (\xi; \theta) \Phi_i^2 (\xi) \right\} + \sum_{j=1}^{M} \sum_{i=1, i \neq j}^{M} C_i (\xi; \theta) C_j (\xi; \theta) \Phi_i (\xi) \Phi_j (\xi) \right\} - E^2 \left[ \hat{y}^{(p)} (\xi) \right] \tag{A.10}
\]

\[
= \sum_{i=1}^{M} E \left[ C_i^2 (\xi; \theta) \right] \Phi_i^2 (\xi) + \sum_{j=1}^{M} \sum_{i=1, i \neq j}^{M} E \left[ C_i (\xi; \theta) C_j (\xi; \theta) \Phi_i (\xi) \Phi_j (\xi) \right] - E^2 \left[ \hat{y}^{(p)} (\xi) \right].
\]

Refer to Eq. (26),

\[
C_i (\xi; \theta) = \left\{ E \left[ C_i (\xi; \theta) \right] + \delta_i^\xi \right\}^2 = E^2 \left[ C_i (\xi; \theta) \right] + 2\delta_i^\xi E \left[ C_i (\xi; \theta) \right] + \left( \delta_i^\xi \right)^2 \tag{A.11}
\]

\[
C_i (\xi; \theta) C_j (\xi; \theta) = \left\{ E \left[ C_i (\xi; \theta) \right] + \delta_i^\xi \right\} \left\{ E \left[ C_j (\xi; \theta) \right] + \delta_j^\xi \right\} = E \left[ C_i (\xi; \theta) \right] E \left[ C_j (\xi; \theta) \right] + \delta_i^\xi E \left[ C_i (\xi; \theta) \right] + \delta_j^\xi E \left[ C_j (\xi; \theta) \right] + \delta_i^\xi \delta_j^\xi \tag{A.12}
\]

where both \( \delta_i^\xi = 2\delta_i^\xi E \left[ C_i (\xi; \theta) \right] + \left( \delta_i^\xi \right)^2 \) and \( \delta_j^\xi = E \left[ C_j (\xi; \theta) \right] \) are infinitesimals for \( \theta \to \theta^* \). Therefore, Eq. (A.10) can be

\[
\text{Var} \left[ \hat{y}^{(p)} (\xi) \right] = \sum_{i=1}^{M} E \left[ C_i^2 (\xi; \theta) \right] \Phi_i^2 (\xi) + \sum_{j=1}^{M} \sum_{i=1, i \neq j}^{M} E \left[ C_i (\xi; \theta) C_j (\xi; \theta) \Phi_i (\xi) \Phi_j (\xi) \right] - E^2 \left[ \hat{y}^{(p)} (\xi) \right]
\]

\[
= \sum_{i=1}^{M} E \left[ E^2 \left[ C_i (\xi; \theta) \right] + \delta_i^\xi \right] \Phi_i^2 (\xi) + \sum_{j=1}^{M} \sum_{i=1, i \neq j}^{M} E \left[ E \left[ C_i (\xi; \theta) \right] E \left[ C_j (\xi; \theta) \right] + \delta_i^\xi \delta_j^\xi \right] \Phi_i (\xi) \Phi_j (\xi)
\]

\[
= \sum_{i=1}^{M} E \left[ E^2 \left[ C_i (\xi; \theta) \right] \right] + \sum_{j=1}^{M} \sum_{i=1, i \neq j}^{M} E \left[ E \left[ C_i (\xi; \theta) \right] E \left[ C_j (\xi; \theta) \right] \right] \Phi_i (\xi) \Phi_j (\xi)
\]

\[
= \sum_{i=1}^{M} E \left[ \delta_i^\xi \right] \Phi_i^2 (\xi) + \sum_{j=1}^{M} \sum_{i=1, i \neq j}^{M} E \left[ \delta_i^\xi \delta_j^\xi \right] \Phi_i (\xi) \Phi_j (\xi)
\]

\[
= \sum_{i=1}^{M} E \left[ \delta_i^\xi \right] \Phi_i^2 (\xi) + \sum_{j=1}^{M} \sum_{i=1, i \neq j}^{M} E \left[ \delta_i^\xi \Phi_i (\xi) \Phi_j (\xi) \right]
\]

\[
= \sum_{i=1}^{M} E \left[ \delta_i^\xi \right] \Phi_i^2 (\xi) + \sum_{j=1}^{M} \sum_{i=1, i \neq j}^{M} E \left[ \delta_i^\xi \Phi_i (\xi) \Phi_j (\xi) \right]
\]

\[
= \sum_{i=1}^{M} E \left[ \delta_i^\xi \Phi_i^2 (\xi) \right] + \sum_{j=1}^{M} \sum_{i=1, i \neq j}^{M} E \left[ \delta_i^\xi \Phi_i (\xi) \Phi_j (\xi) \right]
\]

\[
= \sum_{i=1}^{M} E \left[ \delta_i^\xi \Phi_i (\xi) \right] + \sum_{j=1}^{M} \sum_{i=1, i \neq j}^{M} E \left[ \delta_i^\xi \Phi_i (\xi) \Phi_j (\xi) \right]
\]

\[
= \sum_{i=1}^{M} E \left[ \delta_i^\xi \Phi_i (\xi) \right] \int_{\xi \in \Omega^d} \Phi_i^2 (\xi) d\Gamma (\xi) + \sum_{j=1}^{M} \sum_{i=1, i \neq j}^{M} E \left[ \delta_i^\xi \right] \int_{\xi \in \Omega^d} \Phi_i (\xi) \Phi_j (\xi) d\Gamma (\xi)
\]

\[
- E^2 \left[ \hat{y}^{(p)} (\xi) \right] \left[ \sum_{i=1}^{M} \sum_{l=1}^{N} \frac{1}{N} \sum_{i=1}^{M} \sum_{i=1, i \neq j}^{M} \left[ \sum_{l=1}^{N} \delta_i^\xi \Phi_i (\xi_l) \Phi_j (\xi_l) \right]
\]

\[
\tag{A.13}
\]

44
where $N$ is a finite positive integer. Due to $\hat{\delta}_i^\xi$ and $\hat{\delta}_i^\xi_j$ ($l = 1, 2, \cdots, N$) are infinitesimals, both $\hat{\delta}_i^\xi \Phi_i^2 (\xi_l)$ and $\hat{\delta}_i^\xi_j \Phi_i (\xi_l) \Phi_j (\xi_l)$ are infinitesimals for $\theta \to \theta^*$. Therefore,

$$\hat{\delta}_i^\xi = \frac{1}{N} \sum_{l=1}^N \hat{\delta}_i^\xi_l \Phi_l^2 (\xi_l) + \sum_{j=1}^M \sum_{l=1, l \neq j} \frac{1}{N} \sum_{l=1}^N \hat{\delta}_i^\xi_j \Phi_i (\xi_l) \Phi_j (\xi_l)$$

(A.14)

is an infinitesimal for $\theta \to \theta^*$. On the basis of Eqs.(A.3), (A.4), (A.8) and (A.14), Eq.(A.13) can be

$$\text{Var} \left[ \hat{g}^{(p)} (\xi) \right] = \sum_{i=1}^M E^2 [C_i (\xi; \theta)] \int_{\xi \in \Omega^d} \Phi_l^2 (\xi) dI (\xi) + \sum_{j=1}^M \sum_{l=1, l \neq j} E [C_i (\xi; \theta)] E [C_j (\xi; \theta)] \int_{\xi \in \Omega^d} \Phi_i (\xi) \Phi_j (\xi) dI (\xi)$$

$$- E^2 \left[ \hat{g}^{(0)} (\xi) \right] + \sum_{i=1}^M \frac{1}{N} \sum_{l=1}^N \hat{\delta}_i^\xi_l \Phi_l^2 (\xi_l) + \sum_{j=1}^M \sum_{l=1, l \neq j} \frac{1}{N} \sum_{l=1}^N \hat{\delta}_i^\xi_j \Phi_i (\xi_l) \Phi_j (\xi_l)$$

$$= \sum_{i=1}^M E^2 [C_i (\xi; \theta)] - \left[ E^2 [C_1 (\xi; \theta)] + \delta_\xi \right] + \hat{\delta}_\xi$$

$$= \sum_{i=2}^M E^2 [C_i (\xi; \theta)] - E^2 [C_1 (\xi; \theta)] + (\hat{\delta}_\xi - \delta_\xi)$$

(A.15)

where $\delta_\xi = \hat{\delta}_\xi - \delta_\xi$ is an infinitesimal for $\theta \to \theta^*$. Thus, in addition to the first adaptive expansion coefficient $C_1 (\xi; \theta)$, the variance of $\hat{g}^{(p)} (\xi)$ approaches the sum of squares of the adaptive expansion coefficients for $\theta \to \theta^*$, i.e.,

$$\left| \text{Var} \left[ \hat{g}^{(p)} (\xi) \right] - \sum_{i=2}^M \left\{ E [C_i (\xi; \theta)] \right\}^2 \right| < \varepsilon_4, \quad (\forall \varepsilon_4 > 0)$$

(A.16)

Appendix B.

On the basis of Eq.(51), the limit of the cost function $F (\theta)$ for $\theta \to \theta^*$ is

$$\lim_{\theta \to \theta^*} \left[ \mathcal{L}_{gd} (\xi^{gd}_l, y_l^{gd}; \theta) + \mathcal{L}_{ce}^1 (\xi^{ce}_l; \theta) + \mathcal{L}_{ce}^{2M} (\xi^{ce}_l; \theta) + \mathcal{L}_{ce}^{\text{var}} (\xi^{ce}_l; \theta) \right] = 0. \quad (B.1)$$

According to Eqs.(39), (46), (47) and (48),

$$\mathcal{L}_{gd} (\xi^{gd}_l, y_l^{gd}; \theta) = \frac{1}{N_{gd}} \sum_{i=1}^{N_{gd}} \sum_{i=1}^M c_i (\xi^{gd}_l; \theta) \Phi_i (\xi^{gd}_l) - y_l^{gd} \geq 0$$

$$\mathcal{L}_{ce}^1 (\xi^{ce}_l; \theta) = E \left[ \hat{g}^{(0)} (\xi^{ce}_l; \theta) \right] - E [c_1 (\xi^{ce}_l; \theta)] \geq 0$$

$$\mathcal{L}_{ce}^{2M} (\xi^{ce}_l; \theta) = \text{Var} \left[ \hat{g}^{(0)} (\xi^{ce}_l; \theta) \right] - \sum_{i=2}^M \left\{ E [c_i (\xi^{ce}_l; \theta)] \right\}^2 \geq 0$$

$$\mathcal{L}_{ce}^{\text{var}} (\xi^{ce}_l; \theta) = \frac{1}{N_{ce}} \sum_{i=1}^{N_{ce}} \sum_{i=1}^M c_i (\xi^{ce}_l; \theta) - \frac{1}{N_{ce}} \sum_{i=1}^{N_{ce}} c_i (\xi^{ce}_l; \theta) \geq 0$$

(B.2)

Thus, the following three equations can be obtained by Eq.(B.1), i.e.,

$$\lim_{\theta \to \theta^*} \left[ \mathcal{L}_{gd} (\xi^{gd}_l, y_l^{gd}; \theta) \right] = 0, \quad (B.3)$$
\[
\lim_{\theta \to \theta^*} \left[ L_{cc}^1 (\xi_{ce}; \theta) \right] = 0, \quad (B.4)
\]
\[
\lim_{\theta \to \theta^*} \left[ L_{cc}^{2M} (\xi_{ce}; \theta) \right] = 0, \quad (B.5)
\]
\[
\lim_{\theta \to \theta^*} \left[ L_{ce}^{exr} (\xi_{ce}; \theta) \right] = 0. \quad (B.6)
\]

According to the infinitesimal’s definition, both \( L_{cc}^1 (\xi_{ce}; \theta) \) and \( L_{cc}^{2M} (\xi_{ce}; \theta) \) are infinitesimals for \( \theta \to \theta^* \).

Thus, for \( \theta \to \theta^* \) and \( \forall \varepsilon > 0 \),
\[
L_{cc}^1 (\xi_{ce}; \theta) < \varepsilon \quad (B.7)
\]
\[
L_{cc}^{2M} (\xi_{ce}; \theta) < \varepsilon
\]

Besides, refer to Eqs.\((B.2)\) and \((B.3)\),
\[
\lim_{\theta \to \theta^*} \left[ \frac{1}{N_{gd}} \sum_{l=1}^{N_{gd}} \sum_{i=1}^{M} c_i \left( \xi_{gd}^l; \theta \right) \Phi_i \left( \xi_{gd}^l \right) - y_{gd}^l \right] = 0
\]
\[
\Rightarrow \lim_{\theta \to \theta^*} \left[ \sum_{l=1}^{N_{gd}} \sum_{i=1}^{M} c_i \left( \xi_{gd}^l; \theta \right) \Phi_i \left( \xi_{gd}^l \right) - y_{gd}^l \right] = 0
\]
\[
\Rightarrow \sum_{l=1}^{N_{gd}} \lim_{\theta \to \theta^*} \left[ \sum_{i=1}^{M} c_i \left( \xi_{gd}^l; \theta \right) \Phi_i \left( \xi_{gd}^l \right) - y_{gd}^l \right] = 0, \quad l = 1, 2, \ldots, N_{gd}
\]

Alternatively, Eq.\((B.8)\) can be written in a more generalized form:
\[
\forall \xi \in \Omega^d, \quad \lim_{\theta \to \theta^*} \left[ \sum_{i=1}^{M} c_i (\xi; \theta) \Phi_i (\xi) - y \right] = 0. \quad (B.9)
\]

According to the infinitesimal’s definition, \( \sum_{i=1}^{M} c_i (\xi; \theta) \Phi_i (\xi) - y \) is an infinitesimal for \( \theta \to \theta^* \). Therefore, for \( \theta \to \theta^* \), \( \forall \xi \in \Omega^d \), and \( \forall \varepsilon > 0 \),
\[
\sum_{i=1}^{M} c_i (\xi; \theta) \Phi_i (\xi) - y < \varepsilon. \quad (B.10)
\]

Compared with Eq.\((B.10)\), the coefficients \( c (\xi; \theta) = NN (\xi; \theta) = \{ c_1 (\xi; \theta), c_2 (\xi; \theta), \ldots, c_M (\xi; \theta) \} \) meets the first condition in Eq.\((25)\).
Alternatively, Eq.(B.11) can be written in a more generalized form:

\[
\lim_{\theta \to \theta^*} \left\{ \frac{1}{N_{ce}} - 1 \sum_{i=1}^{M} \sum_{l'=1}^{N_{ce}} \left[ c_i \left( \xi_i^{ce}; \theta \right) - \frac{1}{N_{ce}} \sum_{l'=1}^{N_{ce}} c_i \left( \xi_i^{ce}; \theta \right) \right]^2 \right\} = 0
\]

\[
\Rightarrow \frac{1}{N_{ce}} - 1 \sum_{i=1}^{M} \sum_{l'=1}^{N_{ce}} \lim_{\theta \to \theta^*} \left[ c_i \left( \xi_i^{ce}; \theta \right) - \frac{1}{N_{ce}} \sum_{l'=1}^{N_{ce}} c_i \left( \xi_i^{ce}; \theta \right) \right]^2 = 0
\]

\[
\Rightarrow \frac{1}{N_{ce}} - 1 \sum_{i=1}^{M} \sum_{l'=1}^{N_{ce}} \lim_{\theta \to \theta^*} \left[ c_i \left( \xi_i^{ce}; \theta \right) - \frac{1}{N_{ce}} \sum_{l'=1}^{N_{ce}} c_i \left( \xi_i^{ce}; \theta \right) \right]^2 = 0
\]

\[
\Rightarrow \lim_{\theta \to \theta^*} \left[ c_i \left( \xi_i^{ce}; \theta \right) - \frac{1}{N_{ce}} \sum_{l'=1}^{N_{ce}} c_i \left( \xi_i^{ce}; \theta \right) \right] = 0, \quad i = 1, 2, \cdots, M
\]

\[
\Rightarrow \lim_{\theta \to \theta^*} \{ c_i \left( \xi_i^{ce}; \theta \right) - E \left[ c_i \left( \xi_i^{ce}; \theta \right) \right] \} = 0, \quad i = 1, 2, \cdots, M.
\]

According to the the infinitesimal’s definition, \( c_i \left( \xi_i; \theta \right) - E \left[ c_i \left( \xi_i; \theta \right) \right] \) \((i = 1, 2, \cdots, M)\) is an infinitesimal for \( \theta \to \theta^* \). Therefore, for \( \theta \to \theta^*, \forall \xi \in \Omega^d, \) and \( \forall \varepsilon > 0, \)

\[
|c_i \left( \xi; \theta \right) - E \left[ c_i \left( \xi; \theta \right) \right]| < \varepsilon.
\]

Compared with Eq.(B.13), the coefficients \( c_i \left( \xi; \theta \right) = \mathcal{N} \mathcal{N} \left( \xi; \theta \right) = \{ c_1 \left( \xi; \theta \right), c_2 \left( \xi; \theta \right), \cdots, c_M \left( \xi; \theta \right) \} \) meets the second condition in Eq.(26).

In summary, the coefficients \( c_i \left( \xi; \theta \right) = \mathcal{C} \left( \xi; \theta \right) = \{ c_1 \left( \xi; \theta \right), c_2 \left( \xi; \theta \right), \cdots, c_M \left( \xi; \theta \right) \} \) are the adaptive expansion coefficients of the Deep aPCE model. Therefore, the MLP model \( \mathcal{N} \mathcal{N} \left( \xi; \theta \right) \) \((\theta \to \theta^*)\) can construct the function \( \mathcal{C} \left( \xi; \theta \right) = \mathcal{F} \left( \xi; \theta \right) \) to determine the adaptive expansion coefficients \( \mathcal{C} \left( \xi; \theta \right) = \left\{ c_i \left( \xi; \theta \right) \mid i = 1, 2, \cdots, M \right\} \).

References

[1] Lee, T., Bilionis, I., Tepole, A.B. Propagation of uncertainty in the mechanical and biological response of growing tissues using multi-fidelity gaussian process regression. Computer Methods in Applied Mechanics and Engineering 2020;359:112724. doi:10.1016/j.cma.2019.112724.

[2] Liu, X., Yu, X., Tong, J., Wang, X., Wang, X. Mixed uncertainty analysis for dynamic reliability of mechanical structures considering residual strength. Reliability Engineering and System Safety 2021;209:107472. doi:10.1016/j.ress.2021.107472.

[3] Yao, W., Chen, X., Luo, W., van Tooren, M., Guo, J. Review of uncertainty-based multidisciplinary design optimization methods for aerospace vehicles. Progress in Aerospace Sciences 2011;47(6):450-479. doi:10.1016/j.paerosci.2011.05.001.
[4] Yao, W., Chen, X., Huang, Y., Gurdal, Z., van Tooren, M.. Sequential optimization and mixed uncertainty analysis method for reliability-based optimization. AIAA Journal 2013;51(9):2266–2277. doi:10.2514/1.J052327.

[5] Xu, J., Zhang, Y., Dang, C.. A novel hybrid cubature formula with pearson system for efficient moment-based uncertainty propagation analysis. Mechanical Systems and Signal Processing 2020;140. doi:10.1016/j.ymssp.2020.106661.

[6] Zhang, Y., Xu, J.. Efficient reliability analysis with a cda-based dimension-reduction model and polynomial chaos expansion. Computer Methods in Applied Mechanics and Engineering 2021;373. doi:10.1016/j.cma.2020.113467.

[7] Zheng, X., Yao, W., Xu, Y., Chen, X.. Improved compression inference algorithm for reliability analysis of complex multistate satellite system based on multilevel bayesian network. Reliability Engineering and System Safety 2019;189:123–142. doi:10.1016/j.ress.2019.04.011.

[8] Zheng, X., Yao, W., Xu, Y., Chen, X.. Algorithms for bayesian network modeling and reliability inference of complex multistate systems: Part i – independent systems. Reliability Engineering and System Safety 2020;202. doi:10.1016/j.ress.2020.107011.

[9] Rubinstein, R.Y., Kroese, D.P.. Simulation and the Monte Carlo Method. Wiley Series in Probability and Statistics; 2016. doi:10.1002/9781118631980.

[10] Xiu, D.B., Karniadakis, G.E.. Modeling uncertainty in flow simulations via generalized polynomial chaos. Journal of Computational Physics 2003;187(1):137–167. doi:10.1016/s0021-9991(03)00092-5.

[11] Yao, W., Tang, G., Wang, N., Chen, X.. An improved reliability analysis approach based on combined form and beta-spherical importance sampling in critical region. Structural and Multidisciplinary Optimization 2019;60(1):35–58. doi:10.1007/s00158-019-02193-y.

[12] Oladyshkin, S., Nowak, W.. Data-driven uncertainty quantification using the arbitrary polynomial chaos expansion. Reliability Engineering and System Safety 2012;106:179–190. doi:10.1016/j.ress.2012.05.002.

[13] Wan, H., Ren, W., Todd, M.D.. Arbitrary polynomial chaos expansion method for uncertainty quantification and global sensitivity analysis in structural dynamics. Mechanical Systems and Signal Processing 2020;142. doi:10.1016/j.ymssp.2020.106732.

[14] Tripathy, R.K., Bilionis, I.. Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification. Journal of Computational Physics 2018;375:565–588. doi:10.1016/j.jcp.2018.08.036.

[15] Zhang, D., Lu, L., Guo, L., Karniadakis, G.E.. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. Journal of Computational Physics 2019;397:108850. doi:10.1016/j.jcp.2019.07.048.
[16] Wong Felix, S.. Slope reliability and response surface method. Journal of Geotechnical Engineering 1985;111(1):32–53. doi:10.1061/(ASCE)0733-9410(1985)111:1(32).

[17] Roussouly, N., Petitjean, F., Salaun, M.. A new adaptive response surface method for reliability analysis. Probabilistic Engineering Mechanics 2013;32:103–115. doi:10.1016/j.probengmech.2012.10.001.

[18] Kaymaz, I.. Application of kriging method to structural reliability problems. Structural Safety 2005;27(2):133–151. doi:10.1016/j.strusafe.2004.09.001.

[19] Echard, B., Gayton, N., Lemaire, M.. Ak-mcs: An active learning reliability method combining kriging and monte carlo simulation. Structural Safety 2011;33(2):145–154. doi:10.1016/j.strusafe.2011.01.002.

[20] Li, H.s., Lù, Z.z., Yue, Z.f.. Support vector machine for structural reliability analysis. Applied Mathematics and Mechanics 2006;27(10):1295–1303. doi:10.1007/s10483-006-1001-z.

[21] Bourinet, J.M., Deheeger, F., Lemaire, M.. Assessing small failure probabilities by combined subset simulation and support vector machines. Structural Safety 2011;33(6):343–353. doi:10.1016/j.strusafe.2011.06.001.

[22] Wiener, N.. The homogeneous chaos. American Journal of Mathematics 1938;6.

[23] Ghanem, R., Spanos, P.D.. A stochastic galerkin expansion for nonlinear random vibration analysis. Probabilistic Engineering Mathematics 1993;8(3-4):255–264. doi:10.1016/0266-8920(93)90019-R.

[24] Matthies, H.G., Keese, A.. Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations. Computer Methods in Applied Mechanics and Engineering 2005;194(12-16):1295–1331. doi:10.1016/j.cma.2004.05.027.

[25] Isukapalli, S.S., Roy, A., Georgopoulos, P.G.. Stochastic response surface methods (srsms) for uncertainty propagation: application to environmental and biological systems. Risk analysis : an official publication of the Society for Risk Analysis 1998;18(3):351–363. doi:10.1111/j.1539-6924.1998.tb01301.x.

[26] Li, H., Zhang, D.. Probabilistic collocation method for flow in porous media: Comparisons with other stochastic methods. Water Resources Research 2007;43(9):n/a–n/a. doi:10.1029/2006WR005673.

[27] Shi, L., Yang, J., Zhang, D., Li, H.. Probabilistic collocation method for unconfined flow in heterogeneous media. Journal of Hydrology 2009;365(1-2):4–10. doi:10.1016/j.jhydrol.2008.11.012.

[28] Pan, Q., Dias, D.. Sliced inverse regression-based sparse polynomial chaos expansions for reliability analysis in high dimensions. Reliability Engineering and System Safety 2017;167:484–493. doi:10.1016/j.ress.2017.06.026.

[29] Blatman, G., Sudret, B.. An adaptive algorithm to build up sparse polynomial chaos expansions for stochastic finite element analysis. Probabilistic Engineering Mechanics 2010;25(2):183–197. doi:10.1016/j.probengmech.2009.10.003.
[30] Blatman, G., Sudret, B.. Adaptive sparse polynomial chaos expansion based on least angle regression. Journal of Computational Physics 2011;230(6):2345–2367. doi:10.1016/j.jcp.2010.12.021.

[31] Schmidhuber, J.. Deep learning in neural networks: An overview. Neural Networks 2015;61:85–117. doi:10.1016/j.neunet.2014.09.003.

[32] Goodfellow, I., Bengio, Y., Courville, A.. Deep Learning. The MIT Press; 2016.

[33] Hornik, K., Stinchcombe, M., White, H.. Multilayer feedforward networks are universal approximators. Neural Networks 1989;2(5):359–366. doi:10.1016/0893-6080(89)90020-8.

[34] Cybenko, G.. Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems 1989;2:303–314.

[35] Shahane, S., Aluru, N.R., Vanka, S.P.. Uncertainty quantification in three dimensional natural convection using polynomial chaos expansion and deep neural networks. International Journal of Heat and Mass Transfer 2019;139:613–631. doi:10.1016/j.ijheatmasstransfer.2019.05.014.

[36] Czél, B., Woodbury, K.A., Gróf, G.. Simultaneous estimation of temperature-dependent volumetric heat capacity and thermal conductivity functions via neural networks. International Journal of Heat and Mass Transfer 2014;68:1–13. doi:10.1016/j.ijheatmasstransfer.2013.09.010.

[37] Lye, K.O., Mishra, S., Ray, D., Chandrashekar, P.. Iterative surrogate model optimization (ismo): An active learning algorithm for pde constrained optimization with deep neural networks. Computer Methods in Applied Mechanics and Engineering 2021;374. doi:10.1016/j.cma.2020.113575.

[38] Zhang, X., Xie, F., Ji, T., Zhu, Z., Zheng, Y.. Multi-fidelity deep neural network surrogate model for aerodynamic shape optimization. Computer Methods in Applied Mechanics and Engineering 2021;373. doi:10.1016/j.cma.2020.113485.

[39] Xu, B., Wang, N., Chen, T., Li, M.. Empirical evaluation of rectified activations in convolution network. ArXiv Preprint 2015;arXiv: 1505.00853v2.

[40] Kingma, D.P., Ba, J.. Adam: A method for stochastic optimization. In: International Conference on Learning Representations. 2015,.

[41] Duchi, J., Hazan, E., Singer, Y.. Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research 2011;12:2121–2159.

[42] Ruder, S.. An overview of gradient descent optimization algorithms. ArXiv preprint 2017;arXiv: 1609.04747v2.

[43] Mircea, G.. Stochastic Calculus: Applications in Science and Engineering. Boston: Birkhauser; 2002.

[44] Meecham, W.C., Siegel, A.. Wiener-hermite expansion in model turbulence at large reynolds numbers. Physics of Fluids 1964;7(8):1178–1190.
[45] Siegel, A.. Wiener-hermite expansion in model turbulence in the late decay stage. Journal of Mathematical Physics 2004;6(5):707–721.

[46] Choi, S.K., Grandhi, R.V., Canfield, R.A.. Structural reliability under non-gaussian stochastic behavior. Computers and Structures 2004;82(13-14):1113–1121. doi:10.1016/j.compstruc.2004.03.015.

[47] Nair, V., Hinton, G.. Rectified linear units improve restricted boltzmann machines vinod nair. In: Proceedings of the 27th International Conference on Machine Learning. 2010, p. 1–8.

[48] Sun, Y., Wang, X., Tang, X.. Deeply learned face representations are sparse, selective, and robust. ArXiv preprint 2014;arXiv: 1412.1265.

[49] Baydin, A.G., Pearlmutter, B.A., Radul, A.A.. Automatic differentiation in machine learning: a survey. ArXiv preprint 2015;arXiv: 1502.05767.

[50] Singh, P.K., Jain, P.K., Jain, S.C.. A genetic algorithm-based solution to optimal tolerance synthesis of mechanical assemblies with alternative manufacturing processes: focus on complex tolerancing problems. International Journal of Production Research 2007;42(24):5185–5215. doi:10.1080/00207540410001733931.

[51] Lee, S.H., Chen, W.. A comparative study of uncertainty propagation methods for black-box-type problems. Structural and Multidisciplinary Optimization 2009;37(3):239–253. doi:10.1007/s00158-008-0234-7.

[52] Lee, S.. Reliability based design optimization using response surface augmented moment method. Journal of Mechanical Science and Technology 2019;33(4):1751–1759. doi:10.1007/s00158-019-0327-9.

[53] Fan, W., Wei, J., Ang, A.H.S., Li, Z.. Adaptive estimation of statistical moments of the responses of random systems. Probabilistic Engineering Mechanics 2016;43:50–67. doi:10.1016/j.probengmech.2015.10.005.

[54] Rackwitz, R.. Reliability analysis—a review and some perspectives. Structural Safety 2001;23(4):365–395. doi:10.1016/S0167-4730(02)00009-7.